HOME E PERSONAL COMPUTER

Clive Prigmore

64 Personal Computer e C64

CORSO DI BASIC E GEOS





GRUPPO EDITORIALE JACKSON

Clive Prigmore

64 Personal Computer e C64

CORSO DI BASIC E GEOS



GRUPPO
EDITORIALE
JACKSON
Via Rosellini, 12
20124 Milano

Titolo originale: 30 Hour BASIC Commodore 64

COPERTINA: Silvana Corbelli
GRAFICA E IMPAGINAZIONE: Francesca Di Fiore
FOTOCOMPOSIZIONE: Lineacomp S.r.l. - Milano
STAMPA: Stabilimento Grafico A. Matarelli - Milano

SOMMARIO

PREFAZIONE

Questo libro è destinato agli acquirenti del nuovo COMMODORE 64 poiché insegna sia a programmare in Basic che ad utilizzare il sistema operativo Geos.

Non si presuppone alcuna conoscenza preliminare del sistema, e di conseguenza la materia è stata introdotta molto semplicemente.

Nella prima parte viene presentato un corso di circa 30 ore che vi permetterà di conoscere il linguaggio di programmazione basic, ormai universalmente diffuso.

La seconda parte è dedicata ad insegnare l'uso del dischetto "GEOS V1.0". Questo contiene il sistema operativo GEOS che soddisferà molte esigenze avanzate (disegno computerizzato, elaborazioni di testi,....) altrimenti non ottenibili usando il solo linguaggio BASIC.

Questi tipi di applicazioni necessitano di una interazione uomo macchina molto semplice; Geos soddisfa in pieno a tale esigenza utilizzando principalmente il joystick.

Le due parti possono essere lette indipendentemente l'una dall'altra.

PARTE 1

BASIC

COME USARE QUESTO CORSO

Per utilizzare con facilità e successo un personal computer occorre conoscere tre cose: (a) il linguaggio BASIC; (b) come realizzare delle buone strutture di programma; (c) la tastiera. Questo corso vi insegna principalmente le prime due. Il vostro calcolatore vi insegnerà a usare la terza!

Il testo non fa conoscere tutto sul BASIC, ma espone l'essenziale mettendovi in condizione, alla fine del corso, di realizzare autonomamente dei programmi e, eventualmente di perfezionare la conoscenza del BASIC leggendo altri manuali.

Il corso può essere svolto nel modo seguente:

– *con il computer*, fate tutti gli esercizi e le domande di valutazione (DAV) ed inserite nel computer i programmi consigliati;
– *senza il computer*, fate tutti gli ESERCIZI e le DAV.

Struttura del corso

Il corso è costituito da 10 Unità, ed ogni Unità comprende:

Esempi: sono problemi risolti;

Domande di autovalutazione (DAV): sono domande per controllare se avete capito un nuovo concetto; le risposte vengono date alla fine di ogni Unità;

Esercizi: sono problemi che vi si chiede di svolgere; le soluzioni sono date alla fine di ogni Unità;

UNITA' 1
Le istruzioni e i comandi più semplici

1.1 Cosa fa un computer?

In termini molto generali, un computer è una macchina che ci aiuta a risolvere alcuni tipi di problemi. Si tratta usualmente di problemi che vengono posti facendo uso di simboli o caratteri a noi familiari, per es. lettere dell'alfabeto (maiuscole o minuscole), numeri, segni di punteggiatura e altri caratteri speciali come +, #, *.
Possiamo inserire nel computer un insieme di questi simboli o caratteri e ottenerne un insieme differente ma correlato.
Questa spiegazione ti sembrerà forse molto vaga e troppo generica, perciò è opportuno passare a considerare alcuni casi specifici. Nella tabella che segue, a sinistra puoi vedere alcuni esempi di caratteri che possono essere immessi nel computer e a destra puoi vedere i caratteri che il computer ci dà come risultato.

Caratteri immessi	**Caratteri risultanti**
Dimensioni di una finestra	Costo dell'installazione di un doppio vetro
Elenco di libri presi in prestito da una biblioteca	Lista dei libri da resti–tuire
Nome di una persona	Numero di telefono di quella persona
Regola per una mossa nel gioco degli scacchi	Rappresentazione della scacchiera con la mossa corrispondente
Numero che rappresenta peso e accelerazione	Rappresentazione di un atterraggio lunare
Codici pre–determinati	Suoni musicali

Figura 1 – Alcuni degli usi di un computer.

Questo corso non ti dice come il computer fa queste cose, ma spiega come puoi dirgli tu di farle, dandogli le istruzioni giuste. Non entreremo perciò nel dettaglio delle parti interne di un computer. Tuttavia può essere utile sapere almeno quali sono le parti principali. Ne parleremo rapidamente nel prossimo paragrafo.

1.2 Cos'è un computer

Ti mostriamo subito, nella figura 2, un semplice modello di computer.

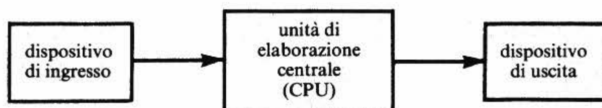


Figura 2 – Un modello semplice di computer.

Come puoi vedere, un computer è costituito da tre parti principali:

1. il dispositivo di ingresso dei dati che ti permette di immettere istruzioni o dati (informazioni) nel computer. Su un microcomputer il dispositivo di ingresso è costituito da una tastiera simile a quella di una macchina da scrivere;

2. l'unità di elaborazione centrale, detta CPU, (dall'inglese Central Processing Unit), che, tra le altre cose, elabora le istruzioni che hai immesso. Si tratta di un'elaborazione che consiste in una modifica dei tuoi dati e che ti dà il risultato o la risposta che richiedi;

3. il dispositivo di uscita che ti permette di ricevere i risultati dell'elaborazione. Il dispositivo di uscita può essere uno schermo televisivo su cui appare il risultato o una stampante che lo scrive su fogli di carta.

Tutto questo può suonare molto vago. Ma ti sarà subito chiaro cos'è un calcolatore, appena saprai quali sono le sue tre caratteristiche principali:
a) la capacità di memorizzare quantità veramente enormi di dati che b) può elaborare molto rapidamente e c) la capacità di immagazzinare un programma che controlli il proprio funzionamento.
Quest'ultima caratteristica è quella che andremo a trattare in questo corso.

Supporti di memoria

Vorremmo accennare ad un altro dettaglio tecnico, prima di passare alla programmazione. Il microcomputer che hai o che stai usando probabilmente avrà una capacità limitata di memoria interna. Esso possiede una specie di deposito, di magazzino in cui sono conservate le principali istruzioni di elaborazione e i dettagli del problema che sta risolvendo.

Questi ultimi dettagli vengono cancellati quando spegni la macchina, perciò, se vuoi conservare il tuo programma e i tuoi dati, devi copiarli su supporti di memoria: un sistema separato di immagazzinamento che puoi collegare al computer, quando è necessario.

Su piccoli sistemi, il supporto di memoria può essere un comune nastro di audiocassetta e su grandi sistemi può essere un disco magnetico. Il Commodore 64 usa un registratore speciale (modello C2N) e non è possibile usare un registratore comune. Per riassumere, gli elementi principali di un calcolatore sono illustrati nella figura 3.

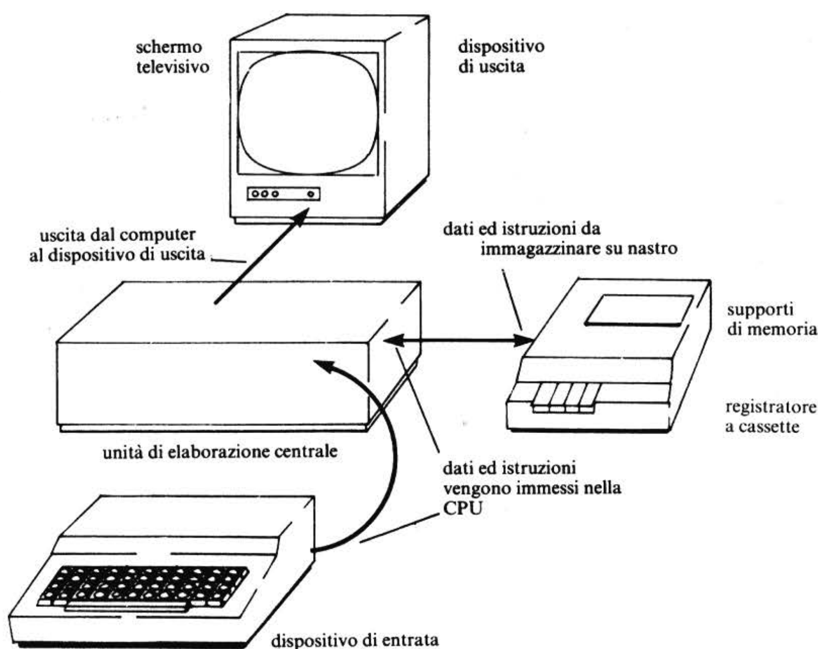


Figura 3 – Un tipico sistema di computer.

1.3 Cos'è il BASIC?

Il computer è un dispositivo elettronico che elabora serie di segnali elettrici. Ma se avessi un problema, non saresti capace di introdurlo nell'elaboratore sotto forma di segnali elettrici. Nè, se il computer fosse pronto a darti un risultato sotto forma di segnali elettrici, saresti in grado di comprenderlo.
Il computer, perciò, ha dentro di sè un CODICE MACCHINA (inserito dal costruttore) che gli permette di capire un CODICE DI PROGRAMMAZIONE che anche tu puoi comprendere.
I codici–macchina sono chiamati LINGUAGGI DI PROGRAMMAZIONE A BASSO LIVELLO e corrispondono direttamente alla serie di segnali elettrici all'interno del computer. Per ovvie ragioni, questo programma è chiamato INTERPRETE. Questo corso ti insegnerà il BASIC, che è un LINGUAGGIO DI PROGRAMMAZIONE AD ALTO LIVELLO. Sarai quindi capace di usare il BASIC per programmare ogni computer che contenga un interprete BASIC (BASIC sta per BEGINNERS' ALL–PURPOSE SYMBOLIC INSTRUCTION CODE, codice di istruzione simbolica valido a tutti gli scopi, per principianti).

Ti sarà utile notare la sequenza di eventi che ha luogo quando programmi un computer:

1. Hai un problema
2. Lo suddividi in passi che possano essere tradotti in BASIC
3. Scrivi il programma in BASIC
4. Ti siedi di fronte alla tastiera e immetti il programma nel computer
5. Il computer interpreta le tue istruzioni BASIC nel proprio codice e le elabora
6. Il computer stampa i risultati nella forma che hai indicato nel programma.

Questo è tutto quello che devi sapere riguardo, un computer. D'ora in avanti assumeremo che tutto quello che farai sarà di sottoporre al computer problemi per ottenerne dei risultati. Procediamo ora con un problema semplice.

1.4 Proviamo a risolvere un problema molto semplice

L'attività principale della programmazione è quella di suddividere la soluzione di un problema in passi semplici che possano essere rappresentati da istruzioni di programmazione BASIC.

Immagina di giocare con un bambino e di impersonare la parte di un computer. Il bambino potrebbe darti due numeri e chiederti di dirgli qual è la loro somma. Dopo un po' di tempo, il bambino naturalmente proverà a darti dei numeri grandi che non puoi addizionare a mente, perciò dovrai avere carta e penna a portata di mano. Quello che segue potrebbe essere un esempio di dialogo tipico:

Bambino: Comincia
Tu: Dammi il primo numero
Bambino: 12157
(scrivi questo numero su un pezzo di carta)
Tu: Dammi il secondo numero
Bambino: 7896

(scrivi il secondo numero sul pezzo di carta)
(esegui la somma)
Tu: 20053

Potremmo descrivere il ruolo del computer in questo processo in modo più formale, come segue:

1. Immetti il primo numero
2. Immetti il secondo numero
3. Somma i due numeri
4. Dai il risultato

Figura 4 – I processi dell'elaborazione nel sommare due numeri.

Attraverso questa semplice analogia, siamo arrivati ad una strategia per risolvere il problema.
Parlando in termini generali, le fasi 1 e 2 riguardano l'immissione di numeri nel computer, la fase 3 interessa un'elaborazione nell'unità di elaborazione centrale, mentre la fase 4 interessa il dispositivo di uscita.

Ora, sebbene non abbiamo introdotto ancora alcun elemento di programmazione BASIC, ti mostreremo quale sarebbe la sequenza di risoluzione del problema, se fosse scritta in BASIC.

Esempio 1
Scrivi un programma BASIC per immettere due numeri nel computer e per ottenere la somma.

Soluzione
Abbiamo già lavorato ad una procedura intuitiva per risolvere questo problema, nella figura 4.
Un programma in BASIC avrebbe la seguente forma (*)

In realtà, sul Commodore 64, dobbiamo utilizzare la parola SEC al posto di SECONDO e più avanti, nel paragrafo 1.9, a pag., te ne spiegheremo il motivo. Modifichiamo quindi leggermente il nostro primo programma:

```
10 INPUT PRIMO
20 INPUT SEC
30 LET SOMMA=PRIMO+SEC
40 PRINT SOMMA
50 END
```

Programma 1

N.d.t.: in questo programma trovi delle istruzioni in inglese. Sono parole che userai per tutto il corso e qui ti indichiamo il loro significato in italiano:
INPUT vuol dire IMMETTI (un dato);
LET vuol dire ASSEGNA (un valore ad una variabile);
PRINT vuol dire STAMPA;
END vuol dire FINE (del programma).

DAV1
Arriviamo ora al primo punto del corso in cui ti consigliamo di verificare quanto hai appreso, affrontando questa domanda di autovalutazione (DAV).

Le DAV sono state progettate per aiutarti a scoprire se hai compreso o no. In ogni caso, la risposta ad ogni DAV viene riportata alla fine dell'unità didattica in cui si trova la domanda. Se rispondi bene a tutte le domande, allora passa all'unità successiva.
Se hai fatto qualche sbaglio, ricontrolla per vedere dove hai fatto l'errore.
Scegli dalla lista B le espressioni che completano correttamente le frasi date in A.

(*) Se volessi provare i programmi di questo corso sul tuo microcomputer, potresti trovare delle difficoltà, se la tua macchina non è quella per cui è stato scritto il corso.

LISTA A

1. la CPU...
2. le caratteristiche principali di un sistema di computer sono ...
3. un codice macchina è ...
4. un codice macchina è un esempio di linguaggio ...
5. il BASIC è un esempio di linguaggio ...
6. l'interprete BASIC ...
7. il programma di un computer è ...

LISTA B

(a) di basso livello
(b) di alto livello
(c) conserva dati e istruzioni, controlla la propria elaborazione e controlla il funzionamento dei dispositivi di entrata e di uscita.
(d) una serie di istruzioni o passi PROCEDURALI per la soluzione di un problema specifico
(e) che è capace di memorizzare grandi quantità di dati, è capace di elaborare dati molto rapidamente, è capace di immagazzinare un programma che controlla il proprio funzionamento.
(f) traduce il codice scritto in BASIC in codice macchina
(g) un codice che corrisponde direttamente alla serie di segnali elettrici all'interno del computer.

1.5 La numerazione delle istruzioni: le etichette

Diamo di nuovo un'occhiata, più attenta, al programma 1:

```
10 INPUT PRIMO
20 INPUT SEC
30 LET SOMMA=PRIMO+SEC
40 PRINT SOMMA
50 END
```

Programma 1

Abbiamo detto che un programma è una sequenza di istruzioni. Nel programma 1 ogni linea costituisce un'istruzione, perciò:

10 INPUT PRIMO
è la prima istruzione del programma e

50 END
è l'ultima.

L'Immissione delle istruzioni
Ancora per un po', ci limiteremo ad affrontare un'istruzione per linea. Quando stiamo seduti di fronte ad una tastiera di un microcomputer (Commodore 64), il processo di immissione di un'istruzione è completato solo dopo aver premuto il tasto {RETURN}. Perciò dovrai battere sulla tastiera 10 INPUT PRIMO e premere {RETURN}. Poi, scrivere 20 INPUT SEC e premere {RETURN}, eccetera.

Vedrai apparire sullo schermo: 10 INPUT PRIMO
20 INPUT SEC
30 ....

Avrai notato che ogni linea comincia con un numero. Deve essere un numero intero compreso tra 1 e 9999 (Vic 20) o 0 e 63999 (Commodore 64) e determina l'ordine in cui le istruzioni vengono lette ed elaborate. Tali numeri definiscono la sequenza delle istruzioni e vengono chiamati ETICHETTE.
L'esecuzione delle istruzioni comincia con la linea che ha l'etichetta inferiore e continua in senso crescente fino ad un'istruzione diversa o fino a quando si arriva alla fine del programma (più avanti ti daremo ulteriori chiarimenti sul significato delle espressioni "fino ad un'istruzione diversa" e "fine del programma").

Perchè poi, potresti chiederti, il programma non è scritto così?

```
1 INPUT PRIMO
2 INPUT SEC
3 LET SOMMA=PRIMO+SEC
4 PRINT SOMMA
5 END
```

Programma 2

Perchè no, naturalmente! Il programma eseguirebbe il suo lavoro altrettanto bene. Comunque, come presto scoprirai scrivendo programmi, hai bisogno di una certa flessibilità.
In particolare, dovresti avere l'opportunità di inserire nel programma un'istruzione che puoi aver dimenticato o una che ti permetta di fare un'importante modifica.
Numerare le nostre linee con le etichette 10, 20, 30 e 40, ci lascia 9 linee vuote

tra un'istruzione e l'altra, linee che possono essere usate per correggere o modificare il programma, senza per questo rallentare in alcun modo l'esecuzione.

DAV2
Guarda i numeri di linea dei programmi seguenti e decidi quali programmi daranno la somma corretta di PRIMO e SEC:

(a)
```
11 INPUT PRIMO
59 INPUT SEC
93 LET SOMMA=PRIMO+SEC
401 PRINT SOMMA
500 END
```

(b)
```
23 INPUT PRIMO
32 INPUT SEC
49 LET SOMMA=PRIMO+SEC
40 PRINT SOMMA
50 END
```

(c)
```
10 INPUT PRIMO
20 INPUT SEC
15 LET SOMMA=PRIMO+SEC
40 PRINT SOMMA
50 END
```

(d)
```
100 INPUT PRIMO
200 INPUT SEC
110 LET SOMMA=PRIMO+SEC
190 PRIMO SOMMA
220 END
```

(e)
```
100 INPUT PRIMO
50 INPUT SEC
407 LET SOMMA=PRIMO+SEC
902 PRINT SOMMA
1000 END
```

(f)
```
200 INPUT PRIMO
300 INPUT SEC
350 LET SOMMA=PRIMO+SEC
250 PRINT SOMMA
400 END
```

Programmi 3-8

1.6 L'esecuzione del programma e i comandi

Il comando RUN

Andiamo avanti e facciamo eseguire il nostro programma, prima che ci venga a noia!

```
10 INPUT PRIMO
20 INPUT SEC
30 LET SOMMA=PRIMO+SEC
40 PRINT SOMMA
50 END
```

Programma 1

Che cosa succede? Niente, perchè il computer aspetta che diamo istruzioni al programma come un tutto. Se vuoi eseguire il programma, devi battere RUN (che in italiano potremmo tradurre come ESEGUI). Dovrai perciò aggiungere una nuova linea.

```
10 INPUT PRIMO
20 INPUT SEC
30 LET SOMMA=PRIMO+SEC
40 PRINT SOMMA
50 END
RUN
```

Programma 1

Non preoccuparti se RUN non ha un'etichetta, ti spiegheremo subito il perchè. Premi il tasto {RET} (o {RETURN} su alcune macchine). Vedrai apparire sullo schermo il simbolo {?}. In questo modo il computer ti chiede dei dati. Dagli il primo numero e premi un'altra volta {RET}; apparirà un altro {?}, poichè il computer ha bisogno di un secondo numero. Dagli il secondo numero e premi {RET}. Ora dovrebbe apparire la risposta. Ti daremo la nostra versione dell'esecuzione di questo programma:

```
10 INPUT PRIMO
20 INPUT SEC
30 LET SOMMA=PRIMO+SEC
40 PRINT SOMMA
50 END
RUN
? 12157
```

```
? 7896
20053

READY.
```

Figura 5 – L'esecuzione completa di un programma.

Stiamo cercando di distinguere tra l'immissione di un programma e la sua esecuzione. Torniamo indietro al dialogo tra il bambino e te che giochi a fare il computer. Un bambino molto sveglio potrebbe aver detto: "voglio darti due numeri e voglio che tu li scriva, che li addizioni e dopo voglio che tu mi dica qual è la loro somma". A questo punto sai esattamente cosa fare, ma non hai fatto ancora niente. Hai avuto le istruzioni, sei stato programmato. Se il dialogo fosse:

Bambino: Comincia
Tu: dammi il primo numero
Bambino: 12157
Tu: dammi il secondo numero
Bambino: 7856
Tu: 20053

le istruzioni ora sarebbero state eseguite. Un programma è perciò proprio una serie di istruzioni per il computer.
Possiamo dire che quando il programma "gira", le istruzioni vengono eseguite.

Altri comandi: LIST, SAVE, LOAD

RUN non è l'unico comando che tratta il programma come un tutt'uno. Puoi usare anche i comandi LIST (che in italiano potremmo tradurre con ELENCA), SAVE (CONSERVA) e LOAD (CARICA).

LIST

Per esempio, ti potrà capitare di perdere molto tempo a battere sulla tastiera un programma e di dover fare parecchie correzioni. Tuttavia, se vuoi vedere una bella copia del programma sullo schermo, puoi battere la parola LIST. Apparirà una copia completa del programma con le linee in ordine di etichetta.

SAVE

Nel caso tu voglia salvare su un supporto magnetico il programma che hai digitato, SAVE è ciò che fa per te (il tuo computer deve essere collegato, naturalmente, ad un registratore).

LOAD

In seguito se vuoi utilizzare di nuovo uno dei programmi che hai conservato, il comando LOAD lo ricaricherà dal supporto di memoria sul tuo computer.
Parole come LIST, RUN, SAVE e LOAD, che ci permettono di usare il programma come un'entità singola, sono chiamate comandi e sono fornite dall'interprete BASIC (vedi unità 10).
Un comando occupa da solo una linea e generalmente non ha un'etichetta, per es. la parola RUN dopo la linea 50 fa sì che il programma cominci l'esecuzione, ed equivale al comando del bambino "comincia", del dialogo precedente.
Discuteremo dei comandi con maggior dettaglio in seguito, ma i quattro che abbiamo appena visto ci permetteranno di iniziare.

Risposte dal computer

Dopo aver eseguito un comando senza aver trovato errori, l'interprete ne informa l'utente, scrivendo un messaggio sullo schermo:

READY

Parole–chiave

Sono quelle parole BASIC con le quali specifichiamo una determinata azione da eseguire. Nei programmi precedenti, abbiamo incontrato INPUT, PRINT e LET.

1.7 L'esecuzione del programma e i dati

Avrai capito che abbiamo scritto un programma che sommerà due numeri qualsiasi in modo molto generale. Per il programma è indifferente sapere quali numeri immettiamo quando riceviamo sullo schermo il segnale {?}. Mentre esegue un programma, il computer deve essere capace di ricordarci di

introdurre i numeri necessari al compito che deve svolgere, e quindi deve essere in grado di richiedere dati specifici. Devi imparare a distinguere chiaramente tra il programma –come insieme di istruzioni più o meno generali– e i dati, che sono i numeri da immettere, quando il programma è in esecuzione, per risolvere un problema particolare. Naturalmente, puoi far "girare" lo stesso programma più volte, e con dati diversi, come vedrai più avanti.
Un altro modo di considerare queste istruzioni è quello di pensare ad un arbitro che raduna i corridori all'inizio di una corsa campestre per dare loro una serie di indicazioni: "andate giù a destra del campo fino all'angolo successivo, arrivate al limite della strada e girate a sinistra giù per il sentiero ...". Le istruzioni dell'arbitro sono analoghe ad un programma. Se i corridori comprendono ciò che l'arbitro sta dicendo, sanno cosa fare, ma sono ancora alla linea di partenza. Non sono partiti. Questa situazione è analoga a quella del programma immesso nella macchina.
Finalmente l'arbitro dice "via!" e la corsa comincia. Così avviene quando il computer comincia ad eseguire il programma. Estendiamo l'analogia e immaginiamo che la nostra corsa campestre sia una corsa per dilettanti.
Immaginiamo che l'arbitro non dia ai corridori istruzioni sufficienti per completare la corsa, ma che dica qualcosa come: "quando arrivate alla fine del sentiero troverete altre istruzioni appese alla quercia". Le istruzioni dovrebbero essere sufficienti a guidare i corridori fino alla parte successiva del percorso, per es. alla prossima indicazione, e così via fino alla fine della corsa.
Analogamente, durante l'esecuzione di un programma, introduciamo dati ulteriori. Questa analogia può aiutarti a vedere l'importante distinzione tra l'immissione di un programma nella macchina, l'esecuzione del programma e poi l'introduzione di altri dati.

DAV3

Più avanti troverai una stampa che contiene parole chiave, comandi, risposte del sistema e dati. Contiene anche sezioni che riguardano l'immissione, l'esecuzione e il listing (*).

(*) Ricordi il comando LIST? Ti permette di vedere sullo schermo tutto il programma, lo "elenca". Useremo questo termine inglese, ormai diventato comune nel linguaggio del programmatore, per indicare questa "visualizzazione", questa "elencazione" del programma sullo schermo o sulla stampante (n.d.t.).

Identifica quanti più elementi ti è possibile come segue:

segna le parole chiave con K
segna i comandi con C
segna le risposte del sistema con R
segna i dati con D

indica e racchiudi tra parentesi le parti che riguardano l'immissione, l'esecuzione, il listing.

```
10 INPUT PRIMO
20 INPUT SEC
30 LET SOMMA=PRIMO+SEC
40 PRINT SOMMA
50 END
RUN
? -37
? -46
-83

READY.

LIST

10 INPUT PRIMO
20 INPUT SEC
30 LET SOMMA=PRIMO+SEC
40 PRINT SOMMA
50 END
READY.
RUN
? 12.83
? 48.95
61.78

READY.
```

Programma 1 (vedi pag. 7)

Nota: READY è una risposta del computer che, per il momento, non devi classificare.

1.8 INPUT, PRINT e LET

A questo punto avrai capito che il programma è una sequenza di istruzioni e ti abbiamo dato una idea di come lavora ogni istruzione. Avrai anche notato che ognuno dei tre tipi di istruzione usati prima (INPUT, LET e PRINT) corrisponde ad uno dei tre principali dispositivi che costituiscono il sistema del computer (dispositivo di immissione, o come si dice dall'inglese, di input, elaboratore centrale e dispositivo di uscita, o di output). Vedremo ora in dettaglio ogni istruzione.

INPUT

La parola INPUT segnala al computer che, durante l'esecuzione, bisogna immettere un dato attraverso il dispositivo di input. Ricorderai cosa è accaduto quando abbiamo fatto girare il nostro primo programma: dopo il punto interrogativo {?} abbiamo immesso 12157, battuto il tasto {RETURN} per completare la procedura d'immissione e poi abbiamo trovato un altro {?} che richiedeva l'immissione del numero successivo. Cosa è successo al primo dato (12157)? E' stato immagazzinato in una "locazione di memoria" indicata con PRIMO, per essere utilizzato successivamente, durante l'esecuzione del programma.
Hai notato che abbiamo introdotto un nuovo concetto, quello di locazione di memoria? Non preoccuparti se non te lo spieghiamo subito, cercheremo di fartelo comprendere nel contesto. In ogni caso, il paragrafo successivo te ne parlerà dettagliatamente.
La parola PRIMO ha due funzioni principali:
(a) quando viene scritta e successivamente richiamata dal programmatore, gli ricorda che, a questo punto, bisogna dare al programma il primo dato
(b) quando viene scritta nell'istruzione 10 INPUT PRIMO è il nome di una locazione nella memoria del computer.
Perciò 10 INPUT PRIMO significa: immetti un numero nel dispositivo di input ed immagazzinalo nella locazione chiamata PRIMO. A seconda del numero immesso, PRIMO assume valori diversi, pertanto è chiamato VARIABILE.

PRINT

L'istruzione 40 PRINT SOMMA ha quasi l'effetto inverso dalle istruzioni 10 e 20 nel senso che ci permette di ricevere informazioni dalla macchina. Segnala alla macchina di prendere una copia dei contenuti della locazione di memoria chiamata SOMMA e passarla al dispositivo di output che, per la maggior parte degli utenti di questo corso sarà uno schermo televisivo.

Nota che il risultato verrà letteralmente stampato, se al tuo microcomputer è collegata una stampante; ma puoi ancora usare il comando PRINT anche quando il tuo microcomputer è collegato al televisore.

LET

30 LET SOMMA = PRIMO + SEC è un esempio di istruzione di assegnazione. In questo tipo di istruzione ha luogo l'esecuzione. Come puoi vedere, è un miscuglio di nomi (SOMMA, PRIMO, SEC) e operatori aritmetici (= e +). Se la leggi più attentamente dice:
"fa sì che la locazione di memoria SOMMA sia uguale ai contenuti della locazione chiamata PRIMO aggiunta ai contenuti della locazione chiamata SEC".

Comunque, come in molte espressioni matematiche, spesso è più chiaro leggere da destra a sinistra del segno "="; "aggiungi i contenuti della locazione PRIMO ai contenuti della locazione SEC e metti il risultato nella locazione chiamata SOMMA".
Generalmente, l'istruzione di assegnazione ha la forma:
LET nome della locazione di memoria = espressione e significa: trova il valore dell'espressione a destra del segno "=" e mettilo nella locazione di memoria indicata a sinistra del segno "=".

LET ... = ... può essere facilmente confuso con ... = ... nelle equazione matematiche. Vediamo, con un esempio, qual'è la differenza. Supponiamo che tu abbia immagazzinato un numero nella locazione L e che voglia sommare 5 a quel numero.

Scrivi:

LET L = L+5

che, ovviamente, non significa:

L = L+5

poichè non c'è valore per L che renda vera quest'ultima espressione. Significa invece che il computer ha aggiunto 5 al numero che si trova nella locazione di memoria L.

1.9 Locazioni di memoria

Come abbiamo già detto, una delle principali caratteristiche di un sistema di computer è la sua capacità di immagazzinare una grande quantità di dati.

Vediamo ora come il linguaggio BASIC ci permette di attribuire nomi alle locazioni di memoria. Dal nostro primo programma ricorderai che un computer è capace di fare una sola cosa alla volta. E' ovvio perciò che quando raggiungiamo la linea 20 e desideriamo immettere il secondo numero, quello immesso nella linea 10 deve essere stato immagazzinato nella locazione chiamata PRIMO. Possiamo pensare alle locazioni di memoria come fossero cellette di un alveare, in cui distinguiamo l'indirizzo o nome di ogni celletta e il suo contenuto.
Nel nostro modello di possibili locazioni di memoria, abbiamo usato i nomi A, B, C... D'altra parte, nel nostro programma, abbiamo usato parole anche di cinque caratteri per indicare le locazioni, per es. PRIMO, SEC, etc.
Arriviamo alla prima differenza principale tra gli interpreti dei vari microcomputer: alcuni interpreti BASIC permettono una varietà più ampia di nomi di locazione, rispetto ad altri.
Alcune macchine limitano il nome di variabile a pochi caratteri. Altre permettono nomi di variabile più lunghi.
Come nomi di variabile, non possono essere usate le parole–chiave del BASIC. Per es., non potremmo usare LET come nome di variabile, perché è una parola–chiave. Come ricorderai non abbiamo usato SECONDO, come nome di variabile nel programma 1, perché include ON, che è una parola–chiave.
Inoltre, alcuni interprete, anche se permettono di utilizzare nomi di variabile

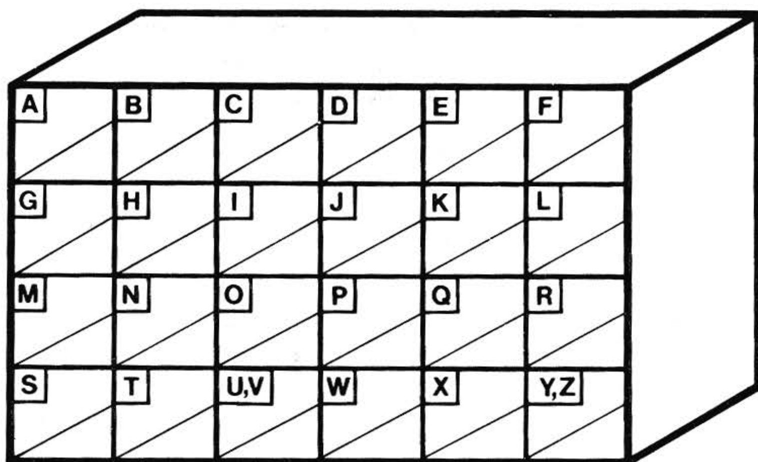


Figura 6 – Un modello di locazioni di memoria in un computer molto piccolo.

lunghi, riconoscono soltanto i primi due caratteri di un nome di variabile. SOMMA, SOLE E SOTTO, per esempio, sono interpretati come fossero lo stesso nome di variabile perché cominciano tutte con le due lettere SO.

La scelta di nomi di locazioni di memoria

Chiaramente, il programmatore avrà vita più facile se sceglie nomi di locazioni di memoria che gli ricordino cosa ha immagazzinato.
Questo è il motivo per cui abbiamo scelto PRIMO, SEC e SOMMA.
Avremmo potuto usare A, B e S; il nostro programma sarebbe stato:

```
10 INPUT A
20 INPUT B
30 LET S=A+B
40 PRINT S
50 END
```

Programma 9

Quando sul libro compare la scritta "- esegui il programma -" significa che ti suggeriamo di farlo per esercitarti. Per far ciò, batti le linee, premi {RET} e poi scrivi RUN.
Il microcomputer ti chiederà un numero, dagliene uno e premi {RET}. Dagli il secondo numero, premi {RET} e la somma apparirà sullo schermo.
Tornando al nostro esempio, dopo aver immesso 12157 e 7896, le locazioni di memoria sarebbero:

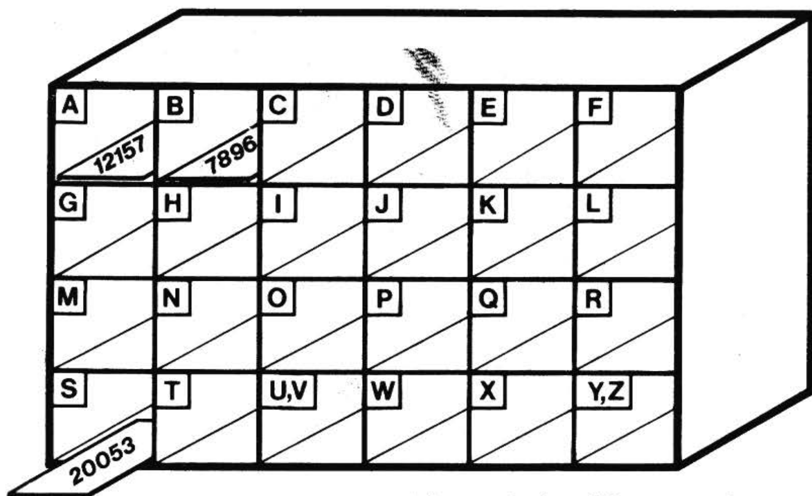


Figura 7 – Stato delle locazioni di memoria dopo il Programma 9.

Il sistema più comune di attribuire nomi di locazioni

Poiché non su tutti i microcomputer è possibile usare i nomi di locazioni di memoria lunghi, d'ora in avanti adotteremo un sistema di nomi di locazioni che, praticamente, lavora su tutti i microcomputer.
Il sistema che useremo chiama, "etichetta", una locazione con una lettera maiuscola o con una lettera maiuscola seguita da una cifra: ci dà quindi 286 possibili locazioni, come illustrato nella figura 8.
Il Commodore 64 richiede che il primo carattere di un nome di locazione di memoria si trovi tra A e Z e il secondo carattere tra Q–Z o 0–9.

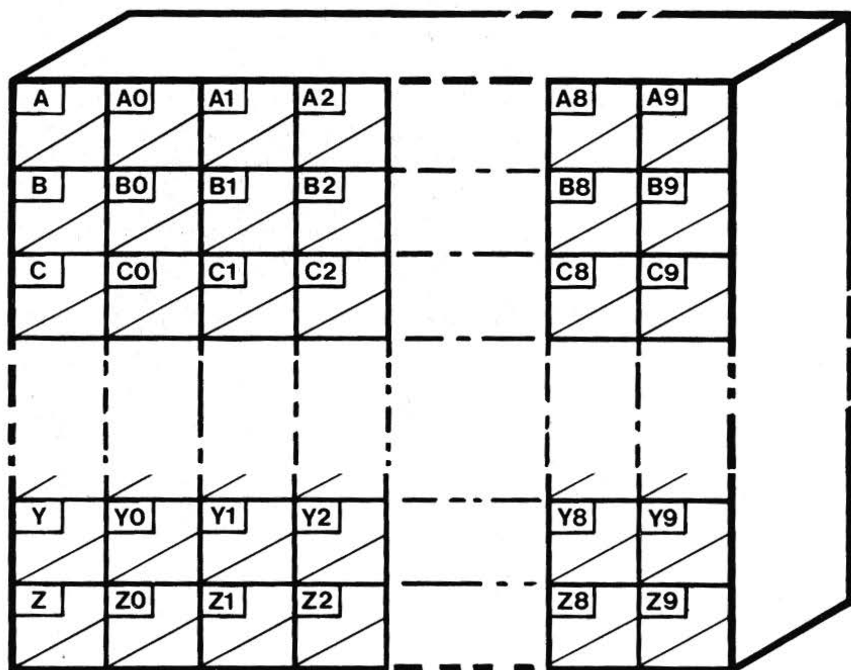


Figura 8 – Le 286 locazioni di memoria possibili.

1.10 COPIA e SOVRASCRITTURA

Le istruzioni BASIC possono avere due effetti differenti, sui contenuti di una

locazione di memoria. O meglio, un'istruzione può non avere effetto sui contenuti della locazione o può cambiarne i contenuti. Vediamo come:

Effetti della copia

Supponiamo di avere il numero 53 nella locazione A. Cosa succede dopo LET B = A e dopo PRINT A? In ogni caso, dopo che l'istruzione è stata eseguita A contiene ancora 53.

Memoria (prima)

53 A	B	C
D	E	F

Istruzione eseguita: LET B = A

Memoria (dopo)

53 A	53 B	C
D	E	F

Memoria (prima)

53 A	B	C
D	E	F

Istruzione eseguita: PRINT A

Memoria (dopo)

53 A	B	C
D	E	F

Schermo del monitor

53

Nei due casi, le istruzioni di copia lasciano inalterate le locazioni di memoria. E' proprio come richiedere un estratto–conto alla tua banca: sul foglio di carta, verrà copiato l'ammontare del tuo conto, ma i soldi rimangono sul conto!

Effetto di sovrascrittura

Supponiamo di avere ancora il numero 53 nella locazione A ma eseguiamo questa volta l'istruzione LET A=A+7. Il risultato è:

Memoria (prima)

53 A	B	C
D	E	F

Istruzione eseguita

LET A = A + 7

Memoria (dopo)

60 A	B	C
D	E	F

L'istruzione LET A=A+7 scrive sul contenuto di A; il contenuto originale cioè sparisce ed è sostituito dal nuovo contenuto che è, in questo caso, 60. Naturalmente, potremmo cambiare in 60 il contenuto di A in altri modi, per es., con LET A=60.

DAV4

Quali dei seguenti sono nomi di locazioni di memoria validi per i numeri secondo le regole date a pag. 20?

(a) N3
(b) 3N
(c) W10
(d) B#
(e) QJ
(f) M
(g) MS
(h) M–5
(i) M+5
(j) U0

Spiega le ragioni per cui hai escluso alcuni dei nomi.

1.11 Gli operatori aritmetici

In aritmetica, si usano quattro operatori principali: +, –, x, :. Il BASIC ha le stesse operazioni, anche se due sono stampate in modo differente:

Simboli aritmetici	Significato	Simboli BASIC
+	somma	+
–	sottrai	–
x	moltiplica	*
:	dividi	/

DAV5

Scrivi le seguenti espressioni aritmetiche usando i simboli del BASIC (dove trovi le parentesi, lasciale anche nelle risposte).

(a) 3+7
(b) 3x7
(c) 8:4
(d) 5x(2+8)
(e) 30:(3+2)
(f) 24–(4x3)
(g) 5x6x7
(h) 81–(27x2)

DAV6

Se A ha valore 2, B ha valore 5 e C ha valore 10, calcola il valore delle seguenti espressioni:

(a) A+B+C
(b) A*B
(c) A*B*C
(d) C/A
(e) C/(B–A)
(f) A*A
(g) (B*C)/(B–A)
(h) (C–B)*(C+B)

In BASIC si fa aritmetica usando istruzioni di assegnazione (istruzioni LET) che dicono all'unità aritmetica del computer (parte dell'unità centrale), cosa fare. Per esemplificare, ricorriamo al seguente modello di computer:

Effetto di LET A=B–C

Memoria all'inizio

A	15 B	10 C
D	E	F

LET A = B–C

Ricorda di leggere da destra a sinistra. L'istruzione dice di prendere il numero dalla locazione B, sottrarre ad esso il numero della locazione C e mettere il risultato nella locazione A.
Il risultato, perciò, è:

Risultato di LET A = B–C

Nota che i contenuti di B e C sono rimasti immutati.

Memoria alla fine

5 A	15 B	10 C
D	E	F

Effetto di LET A = B∗C

Memoria all'inizio

A	15 B	10 C
D	E	F

Memoria alla fine

150 A	15 B	10 C
D	E	F

DAV7

Colloca i valori nelle locazioni di memoria A, B, C dopo l'esecuzione di ogni linea dei seguenti programmi:

1. Programma 1

	Valori delle locazioni di memoria		
	A	B	C
10 LET A = 12	☐	☐	☐
20 LET B = 5	☐	☐	☐
30 LET C = A∗(A+B)	☐	☐	☐
40 LET A = A+10	☐	☐	☐

2. Programma 2

Valori delle locazioni di memoria

	A	B	C
10 LET A = 20	□	□	□
20 LET B = A*3	□	□	□
30 LET C = A/4	□	□	□
40 LET A = B+C	□	□	□

Quello che hai appena fatto non è (speriamo) difficile e non sarà più difficile passare a nomi di locazione di memoria più complicati. Siamo costretti a rendere i nomi un po' più complicati perché A, B, C... ci danno solo 26 locazioni (corrispondenti alle 26 lettere dell'alfabeto inglese) e, come abbiamo detto a pag. 20 useremo i nomi di locazione A, A0, A1, eccetera. Perciò:

LET P4=Q1*R1

non è differente né più difficile di LET A=B*C. P4, Q1 e R1 sono semplicemente nomi di locazioni di memoria. P4 è un nome, proprio come MI34567P è un numero di targa.

Ora siamo pronti ad usare la capacità aritmetica di un computer.

Esempio 2

Scrivi un programma BASIC per immettere due numeri nel computer ed ottenere la loro somma, la differenza, il prodotto e il quoziente.

Soluzione

Il problema può sembrare complicato, ma in realtà lo abbiamo già risolto. Avevamo un programma (il programma 9) che ci dava la somma di due numeri, perciò, per avere la loro differenza, il prodotto e il quoziente dobbiamo cambiare solo l'operatore aritmetico sulla linea 30, che era:

30 LET SOMMA=PRIMO+SEC

Prima, però, riscriviamo il programma usando nomi di memoria più brevi:

Versione originale

```
10 INPUT PRIMO
20 INPUT SEC
30 LET SOMMA=PRIMO+SEC
40 PRINT SOMMA
50 END
```

Versione nuova

```
10 INPUT N1
20 INPUT N2
30 LET S=N1+N2
40 PRINT S
50 END
```

Programma 10

Ora, abbiamo bisogno di altre tre versioni, ognuna con una linea 30 differente:

10 INPUT N1 20 INPUT N2 30 LET S = N1+N2 40 PRINT S	" " " " " D = N1−N2 " D	" " " " " P = N1 * N2 " P	" " " " " Q = N1/N2 " Q

Usiamo D per la locazione differenza, P per la locazione prodotto e Q per la locazione quoziente.

Dobbiamo scrivere quattro programmi? Fortunatamente no, perché quando copiamo i numeri dalle locazioni N1 e N2, non ne distruggiamo i contenuti e perciò possiamo usarli quattro volte in un unico programma.

```
10 INPUT N1       ┐
20 INPUT N2       │
30 LET S=N1+N2    ├──── programma originale
40 PRINT S        ┘     per la somma
50 LET D=N1-N2    ┐
60 PRINT D        ┘──── linea extra per la differenza
70 LET P=N1*N2    ┐
80 PRINT P        ┘──── linee extra per il prodotto
90 LET Q=N1/N2    ┐
100 PRINT Q       ┘──── linee extra per il quoziente
110 END
```

Programma 11 – Somma, differenza, prodotto e quoziente di due numeri.

Esegui il programma 11 sul tuo microcomputer. Poi batti RUN e immetti due numeri. Una stampa tipica sarebbe:

```
RUN
? 57.82
? 19.11
76.93
38
1104.9402
3.02564103

READY.
```

1.12 Le costanti numeriche

Precedentemente, in questa unità didattica, abbiamo visto che il primo programma BASIC poteva lavorare con numeri interi, decimali e negativi. A questo stadio, non ci interessa descrivere come vengono rappresentati in BASIC i numeri, ti mostreremo solo come usare i numeri direttamente nelle istruzioni di assegnazione.
L'istruzione LET P=427*R significa prendere il numero 427 moltiplicarlo per il numero trovato nella locazione R e poi immagazzinare il risultato nella locazione P (non devi pensare che i computer manipolino solo numeri binari: l'interprete ci permette di introdurre comuni numeri decimali). In modo simile, l'istruzione LET Y4=3.142+Z8 prende il numero 3.142, lo aggiunge al contenuto della locazione Z8 e mette la somma nella locazione Y4.
Così, l'istruzione LET A=–4893/B prende il numero –4893, lo divide per il numero contenuto nella locazione B e mette i risultati di questo calcolo nella locazione A.

Preambolo all'esercizio

La trasformazione dal sistema di misura anglosassone in quello metrico–decimale è stata lenta e la vita quotidiana di un inglese abbonda ancora di piccole irritanti conversioni che occasionalmente lo infastidiscono, come ad esempio trasformare libbre in chilogrammi, iarde in metri, pinte in litri, once in grammi, etc. Se andiamo in ferie negli Stati Uniti, convertiamo mentalmente chilometri in miglia, lire in dollari.
Molti misurano ancora le temperature del corpo e del tempo in gradi Fahrenheit piuttosto che in gradi centigradi o Celsius.
Possiamo immaginare un microcomputer casalingo del futuro, con un programma generale di conversione che farà per noi tutte queste diverse trasformazioni. Nei prossimi due esercizi ti chiederemo di scrivere programmi che facciano conversioni. Ti servirà ricordare solo i concetti introdotti nei primi programmi di questa unità didattica.

Esercizio 1

Scrivi tre programmi in Basic, che effettuino le seguenti conversioni:
(a) dato un numero che rappresenta una lunghezza in pollici, converti questa lunghezza in centimetri, tenendo conto che un pollice = 2.54 centimetri.
(b) dato un numero che rappresenta un peso in once, converti questo peso in grammi, tenendo conto che un'oncia = 28.375 grammi.

(c) dato un numero che rappresenta una lunghezza in piedi, converti tale lunghezza in centimetri, tenendo conto che un piede = 30.48 cm.
(Le risposte agli esercizi si trovano alla fine di ogni unità didattica, dopo le risposte alle DAV).

Esercizio 2

Ogni conversione implica "un fattore di conversione x e un numero da convertire". Perciò, è possibile scrivere un programma di conversione generale in cui immettere due numeri: il fattore di conversione e il numero da convertire.
Scrivi il programma di conversione generale.

1.13 L'istruzione REM

L'istruzione REM ci permette di dare un nome ad un programma o fare altre annotazioni significative. Per esempio, ci aiuta ad identificare che cosa fa il programma o una sua parte. L'istruzione REM non viene eseguita dal computer: sta solo a beneficio del programmatore e dell'utente e quando il computer vede REM su una linea, ignora tutto ciò che si trova su quella linea dopo REM. REM è un'abbreviazione della parola inglese "remark", che in italiano – in questo contesto – potremmo tradurre con "annotazione". Il prossimo programma riguarda il calcolo delle percentuali e perciò la prima istruzione del programma sarà:
10 REM **CALCOLO PERCENTUALE**
Gli asterischi (**) non hanno altra funzione se non quella di enfatizzare il nome.

Esempio 3

Scrivi un programma BASIC per immettere due numeri ed emettere il secondo come percentuale del primo.
Ricorda che la regola per il calcolo della percentuale è:
(SEC:PRIMO) X 100.

Soluzione

```
10 REM **CALCOLO PERCENTUALE**
20 INPUT F
30 INPUT S
40 LET P=(S/F)*100
50 PRINT P
60 END
```

(titolo del programma usando REM)

Programma 12 – Calcolo della percentuale

Esecuzioni

```
RUN
? 57
? 74
129.824561

READY.

RUN
? 74
? 57
77.0270271

READY.
```

- ESEGUI IL PROGRAMMA 12 -

In generale, è meglio "pulire lo schermo" da ogni informazione non richiesta, quando gira un programma.
Possiamo farlo introducendo l'istruzione

15 PRINT CHR $ (147)

Il carattere 147 dell'insieme dei caratteri immagazzinati nella memoria del computer significa "pulisci lo schermo".
Per maggiori dettagli, vedi la fine dell'unità 3, in cui useremo questa istruzione all'inizio di tutti i programmi.

1.14 Problemi aritmetici un po' più complessi

Abbiamo raggiunto uno stadio in cui possiamo usare il computer come una semplice macchina calcolatrice a quattro funzioni, ma ci piacerebbe qualcosa di un po' più complicato.

Generalmente, il BASIC ci permette di svolgere le equazioni in modo a noi familiare. Possiamo usare le parentesi, per es. ( ), per raggruppare insieme certi valori e quando il BASIC svolge un'espressione, considera per primi i valori all'interno delle parentesi. Poi passa ad eseguire moltiplicazioni e divisioni ed infine addizioni e sottrazioni.
Quest'ordine di precedenza per l'esecuzione di operazioni aritmetiche è discusso più ampiamente in un'unità didattica successiva, ma vedrai presto che quest'ordine formalizza solo il modo in cui procediamo usualmente con i calcoli aritmetici.
Ti mostriamo subito cosa vogliamo dire.

Esempio 4

Scrivi in BASIC le seguenti espressioni:
1. ab+c 2. a(b+c) 3. a/(b+c)

Soluzioni

1. A*B+C
   Le regole di ordine di precedenza ci dicono che A*B va considerato per primo e poi va aggiunto C. Se non fossi sicuro di quest'ordine, potresti scrivere (A*B)+C, ma qui le parentesi non sono essenziali.
2. A*(B+C)
   Nota che proprio come le parentesi sono necessarie in a(b+c), così sono necessarie in A*(B+C).
3. A/(B+C)
   Ora prova a fare qualcosa da solo.

DAV8

Scrivi in BASIC le seguenti espressioni:

1. abc

2. $\frac{ab}{c}$

3. $\frac{a+b}{c}$

Esercizio 3

Ora che hai scritto in BASIC le espressioni della DAV8, scrivi un programma che ti permetta di dare tre numeri (A, B e C) e ottenere i valori di quelle espressioni.

1.15 La stampa letterale

Hai già visto che possiamo stampare i valori che si trovano nelle locazioni di memoria. Più avanti, in questo corso, scoprirai la versatilità dell'istruzione PRINT. Per esempio, permette di stampare sullo schermo del monitor i messaggi che saranno di aiuto all'utente, quando il programma gira. Questi messaggi generalmente vengono chiamati SEGNALI (o prompt).
Abbiamo già visto che quando si incontra un'istruzione di input durante l'esecuzione di un programma, sullo schermo appare un punto interrogativo {?} che ci ricorda che viene richiesto un input. In programmi leggermente più complicati, una serie di punti interrogativi sullo schermo confonde l'utente, poiché non possiamo sapere quale valore di input il punto interrogativo sta chiedendo. In queste circostanze, i segnali generati dall'istruzione PRINT sono molto utili. E' facile far sì che un computer stampi un messaggio sullo schermo.
Basta una linea come:

```
10 PRINT "MESSAGGIO"
```

che stampa semplicemente MESSAGGIO sul tuo schermo.
In altre parole, qualsiasi cosa appaia tra le virgolette " " dopo la parola PRINT, sarà stampata esattamente come si trova.
Nota che, come nel caso dell'istruzione REM, il computer non elabora le parole tra virgolette " ".
Perciò:

```
20 PRINT "A+B"
```

produce

```
A+B
```

sul tuo schermo e il computer non somma il valore della locazione A al valore della locazione B.

Il seguente esempio dimostra l'uso di PRINT " ", per ricordare al programmatore e all'utente cosa fa il programma.

Esempio 5

Scrivi un programma BASIC per convertire un valore di temperatura dato da gradi centigradi (C) in gradi Fahrenheit (F).

Ricorda che: $$F = \frac{9}{5} xC+32$$

```
10 REM **CONVERSIONE DI TEMPERATURE**
15 PRINT CHR$(147)
17 PRINT"CONVERSIONE TEMPERATURA"
20 PRINT"VALORE DELLA TEMPERATURA IN GRADI C'
30 INPUT C
40 LET F=(9/5)*C+32
50 PRINT"LA TEMPERATURA IN GRADI F E'"
60 PRINT F
70 END
```

Programma 13 – Conversione di temperatura

Esecuzione tipica

```
RUN
CONVERSIONE TEMPERATURA
VALORE DELLA TEMPERATURA IN GRADI C
? 16
LA TEMPERATURA IN GRADI F E'
60.8

READY.
```

- ESEGUI IL PROGRAMMA 13 -

Inserendo 17 Print " " facevamo risaltare ancora di più il titolo sullo schermo.

Compito 1

Per lo studente a distanza: la soluzione a questo compito va spedita al tuo

insegnante perché la corregga. Se possiedi un microcomputer, scrivi il compito in BASIC e indica all'insegnante cosa fa il microcomputer.
Completa i compiti secondo le istruzioni che ti sono state date dal tuo centro locale.
Ricorda di usare appropriatamente le istruzioni REM e di stampa letterale quando scrivi i programmi.

1. Se depositi D lire in un conto che ti dà il tasso di interesse in percentuale P per un anno, allora il ricavo alla fine del primo anno è dato dall'equazione:

$$Y = D \times \frac{P}{100}$$

(a) Scrivi un programma BASIC per immettere valori per D e P e dare come risultato il ricavo Y.
(b) Se il deposito originale insieme con l'interesse viene lasciato sul conto per un altro anno allo stesso tasso di interesse, allora l'interesse composto dopo il secondo anno sarà dato dall'equazione:

$$C = (D + Y) \times \frac{P}{100}$$

Estendi il tuo programma (a) per calcolare ed emettere l'interesse composto.

2. Considera il problema della stima del costo di installazione di doppi vetri con infissi in alluminio.
Le finestre sono formate da tre parti:
(a) una cornice di legno
(b) una struttura in alluminio e
(c) il vetro

Se l'altezza della finestra è H e la lunghezza è L metri, allora le lunghezze totali richieste del legno e dell'alluminio sono date approssimativamente dall'espressione (2H + 2L) metri; e l'area del vetro richiesta è data dall'espressione (HxL) metri quadrati.

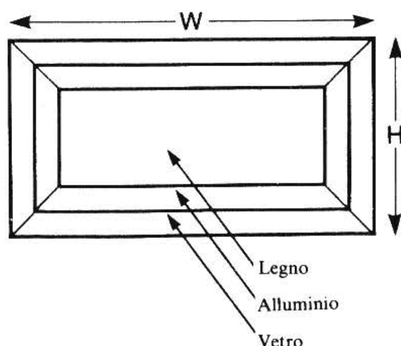


Figura 9

Scrivi tre distinti programmi BASIC che, su valori dati per altezza e lunghezza in metri, daranno il costo di:

(a) il legno della cornice, se il costo del legno è di 3000 per metro
(b) l'alluminio della cornice se il costo dell'alluminio è di 4000 per metro
(c) il vetro se il costo del vetro è di 4000 per metro quadro.

Ora lega queste tre parti in un programma unico per stimare ed emettere il costo totale dell'installazione, se il costo del lavoro è di 50000 per finestra.

Obiettivi dell'unità 1

Ora che hai completato questa unità didattica, controlla se sei capace di:

- scrivere un semplice programma usando:
  Numeri di linea (etichette) ☐
  INPUT ☐
  LET ☐
  PRINT ☐
  Locazioni di memoria identificate da una lettera singola o da una lettera più un numero ☐
  Copia da una locazione ad un'altra ☐
  Sovrascrittura ☐
  +, −, *, / ☐
  ( ) ☐
  Costanti numeriche ☐
  REM ☐
  PRINT " " ☐

- usare:
  RETURN ☐
  RUN ☐

- rispondere a:
  READY ☐
  ?, > ecc. ☐

Risposte alle domande di autovalutazione e agli esercizi (DAV)

DAV1

A	B
1	(c)
2	(e)
3	(g)
4	(a)
5	(b)
6	(f)
7	(d)

DAV2

(a) e (e) eseguono il programma 1. In (b) viene chiesto di stampare SOMMA prima che la somma sia stata calcolata. In (c) e (d) viene chiesto di calcolare SOMMA prima di introdurre SEC.

DAV3

```
        (K)
     10 INPUT PRIMO                  ]
     20 INPUT SEC                    ]  emissione
     30 LET SOMMA=PRIMO+SEC          ]
     40 PRINT SOMMA                  ]
C—   50 -END
C—   RUN
R—   ?-37  ------------------- D     ]
R—   ?-46  ------------------- D     ]  esecuzione
     -83   ------------------- R     ]

R—   READY (K)
C—   LIST
     10 INPUT PRIMO                  ]
     20 INPUT SEC                    ]  listing
     30 LET SOMMA=PRIMO+SEC          ]
     40 PRINT SOMMA                  ]
K—   50 END

R—   READY
C—   RUN
```

```
R— ?12.83   ──────────────── D  ┐
R— ?43.95   ──────────────── D  │ esecuzione
   61.78    ──────────────── R  ┘
R— READY
   RUN
C— ?17.0009 ──────────────── D  ┐
R— ?-29.2629 ─────────────── D  │ esecuzione
R—  -12.362 ──────────────── R  ┘
R— READY
```

(hai notato che questo programma contiene numeri negativi e decimali frazionari?)

DAV4

(a) OK
(b) no, comincia con una cifra invece che con una lettera
(c) non è permesso su tutti i sistemi (10 contiene due cifre, non una sola cifra come W8)
(d) no, su Commodore 64, # non è simbolo accettabile in un nome di variabile
(e) no, usa due lettere (su alcune macchine, però può funzionare). Ok sul Commodore 64.
(f) OK
(g) OK
(h) no, "–" non è un simbolo accettabile
(i) no, "+" non è un simbolo accettabile
(j) OK

DAV5

(a) 3+7 (b) 3*7 (c) 8/4 (d) 5*(2+8) (e) 30/(3+2) (f) 24–(4*3) (g) 5*6*7 (h) 81–(27*2)

DAV6

(a) 17 (b) 10 (c) 100 (d) 5 (e) 10/3 o 3 1/2 o 3.33 ...
(f) 4 (g) 50/3 o 16 2/3 o 16.66 ... (h) 75

DAV7

1.

12		
12	5	
12	5	204
22	5	204

2.

20		
20	60	
20	60	5
65	60	5

Esercizio 1

(a) Programma 14

```
10 INPUT L1
20 LET L2=2.54*L1
30 PRINT L2
40 END
```

Esecuzioni tipiche

```
RUN
? 12
30.48          — primo uso

READY.
RUN
? 36
91.44          — secondo uso

READY.
```

- ESEGUI IL PROGRAMMA 14 -

(b) Programma 15

```
10 INPUT W1
20 LET W2=W1*28.375
30 PRINT W2
40 END
```

Esecuzioni tipiche

```
RUN
? 10
283.75         — primo uso

READY.
RUN
?50
1418.75        — secondo uso

READY.
RUN
? 16
454            — terzo uso

READY.
```

- ESEGUI IL PROGRAMMA 15 -

Esercizio 2

```
10 INPUT V
20 INPUT F
30 LET N=F*V
40 PRINT N
50 END
```

Esecuzioni tipiche

```
RUN
? 16
? 28.375
454

READY.
```

Trasformazione da once in grammi

```
RUN
? 36
? 2.54
91.44

READY.
```

Trasformazione da pollici in centimetri

- ESEGUI IL PROGRAMMA 16 -

DAV8

1. A*B*C
2. A*B/C*(A*B)/C]
3. (A+B)/C

Programma 17

```
4 REM *ARITMETICA*
6 PRINT CHR$(147)
8 PRINT"INSERISCI TRE NUMERI"
10 INPUT A
20 INPUT B
30 INPUT C
40 LET R=A*B*C
50 PRINT R
60 LET R=(A*B)/C
70 PRINT R
80 LET R=(A+B)/C
90 PRINT R
100 END
```

(righe 30–50) calcola la prima espressione e la stampa
(righe 60–70) calcola la seconda espressione e la stampa
(righe 80–90) calcola la terza espressione e la stampa

Nota che possiamo usare R come locazione per tutte le risposte perché stampiamo una risposta alla volta prima di coprirla con la risposta successiva.

Esecuzioni tipiche

```
RUN
INSERISCI TRE NUMERI
? 13
? -27
? 55.2
 -19375.12
 -6.35869595
 -.253623188

READY.
```

```
RUN
INSERISCI TRE NUMERI
? 13
? 13
? 13
 2197
 13
 2

READY.
```

- ESEGUI IL PROGRAMMA 17 -

UNITÀ 2
Prendere decisioni

2.1 Introduzione

I programmi che abbiamo svolto nella prima unità didattica erano abbastanza semplici. L'esecuzione cominciava dall'istruzione con l'etichetta di valore inferiore e continuava in ordine di numero di linea fino al completamento del programma, e cioè fino alla linea con l'etichetta più alta.
I computer sanno fare bene una cosa: i calcoli ripetitivi.
Un'altra cosa che sanno fare bene è prendere delle decisioni. Entrambe queste caratteristiche interessano il cambiamento della sequenza in cui viene eseguito il programma. Questa unità didattica introduce qualche istruzione che ti permetterà di scrivere programmi di questo tipo. Ma prima, introdurremo un nuovo tipo di istruzione PRINT.

2.2 PRINT ...,

Nella prima unità didattica abbiamo scritto un programma (esempio 3) per ricavare un numero come percentuale di un altro. Sullo schermo, il calcolo e il risultato sono apparsi nel formato:

```
RUN
? 57
? 74
129.824561

READY.
```

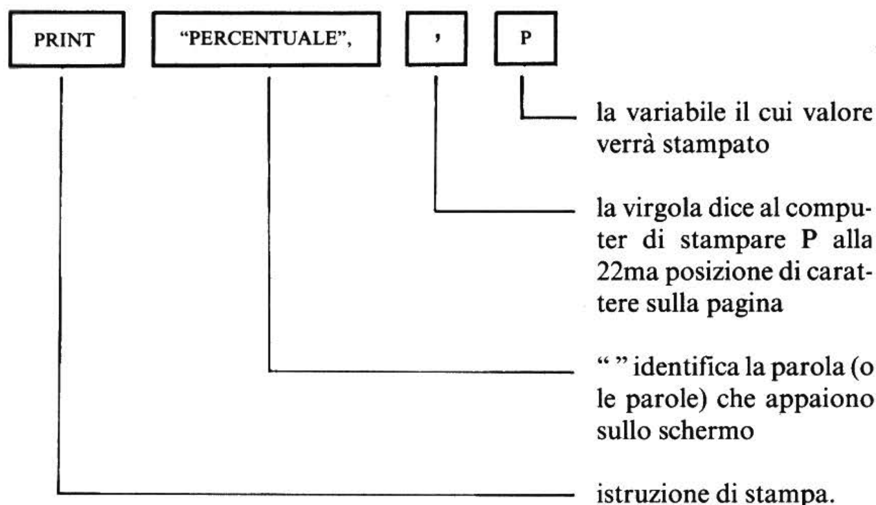
Ovviamente sarebbe stato meglio se la risposta includesse la parola PERCENTUALE in modo che fosse più chiaro cosa sta succedendo. Si può fare facilmente cambiando la linea 50, da PRINT P in:

```
50 PRINT"PERCENTUALE",P
```

L'effetto è:

Versione della linea 50	Risultato sullo schermo
`50 PRINT P`	`129.824561`
`50 PRINT"PERCENTUALE",P`	`PERCENTUALE 129.824561`

L'istruzione PRINT "PERCENTUALE", P ha quattro parti:



Il computer stampa il valore di P in modo che l'ultima cifra (all'estrema destra) si trovi in colonna 31 e il numero viene disposto sulle colonne, da destra a sinistra.
Possiamo usare PRINT ..., per migliorare il programma di calcolo di percentuale dell'Unità didattica 1. Allo stesso tempo, possiamo migliorare l'aspetto del programma facendo uso dell'istruzione di stampa letterale PRINT " ", che abbiamo introdotto nel paragrafo 1.15.

Programma originale

```
10 REM ** CALCOLO PERCENTUALE **
20 INPUT F
30 INPUT S
40 LET P=(S/F)*100
50 PRINT P
60 END
```

Nuovo programma

```
10 REM **CALCOLO PERCENTUALE**
15 PRINT CHR$(147)
20 PRINT"CALCOLO PERCENTUALE"
30 PRINT"INSERISCI IL PRIMO NUMERO"
40 INPUT F
50 PRINT"INSERISCI IL SECONDO NUMERO"
60 INPUT S
70 LET P=(S/F)*100
80 PRINT"PERCENTUALE",P
90 END
```

Programma 1 – Programma di calcolo percentuale migliorato

Programma originale: esecuzione tipica

```
RUN
? 80
? 37
46.25
```

Nuovo programma: esecuzione tipica

```
RUN
CALCOLO PERCENTUALE
INSERISCI IL PRIMO NUMERO —— effetto della linea 20 inserisci
? 80
INSERISCI IL SECONDO NUMERO — effetto della linea 40 inserisci
? 37
PERCENTUALE          46.25  — effetto della linea 70
                     ||
READY.               |└──── 22ma posizione
                     |
                     └── 21ma posizione riservata al segno "–"
                         davanti alle cifre
```

Nota come l'uso della stampa letterale alle linee 20 e 40, insieme con PRINT " ", rende il programma molto più comprensibile.

- ESEGUI IL PROGRAMMA 1 -

Le virgole nell'istruzione PRINT

Nella istruzione di stampa puoi usare più virgole per spaziare i tuoi risultati sullo schermo.
Il Commodore 64 ha quattro zone di stampa sullo schermo:

40 CARATTERI

Zona 1	Zona 2	Zona 3	Zona 4
da 0 a 9	da 10 a 19	da 20 a 29	da 30 a 39

Ogni virgola successiva, nella linea di stampa, sposta il testo tra virgolette verso la zona alla sua destra.

PRINT "ZONA 1", "ZONA 2", "ZONA 3"

risulta:

ZONA 1   ZONA 2   ZONA 3

e

PRINT "PERCENTUALE", P

ha dato

PERCENTUALE   46.25

laddove

PRINT "PERCENTUALE"; P

avrebbe dato

PERCENTUALE 46.25

DAV 1

Cosa apparirebbe sullo schermo come risultato di queste linee di stampa? (Assumi che A = 48, B = 8, C = 6)

(a) PRINT "AREA" A
(b) PRINT "BASE" B, "ALTEZZA" C, "AREA" A
(c) PRINT "BASE", "ALTEZZA", "AREA"
PRINT B, C, A

Scrivi le linee PRINT, in BASIC, per stampare sullo schermo le seguenti parole nelle zone indicate:

	Zona 1	Zona 2	Zona 3	Zona 4
(d)	BASE	8	ALTEZZA	6
(e)	BASE	8		
	ALTEZZA	6		
	AREA	48		
(f)	BASE	ALTEZZA		AREA

2.3 Ripetizioni e GOTO

Supponiamo ora che tu voglia usare il Programma 1 per calcolare tutte le percentuali –rispetto al punteggio massimo ottenibile– dei punteggi ottenuti ad un test somministrato ad una classe di alunni. Supponiamo che il punteggio massimo sia 80. Dovresti usare il programma tante e tante volte, cominciando ogni volta da RUN, come ad es:

```
RUN
CALCOLO PERCENTU•OO
INSERISCI IL PRIMO NUMERO
? 80
INSERISCI IL SECONDO NUMERO
? 42
PERCENTUALE          52.5

READY.

RUN
CALCOLO PERCENTUALE
INSERISCI IL PRIMO NUMERO
? 80
INSERISCI IL SECONDO NUMERO
? 19
PERCENTUALE          23.75

READY.
```

Figura 1 – Uso ripetuto del programma di calcolo percentuale.

Sarebbe molto più facile se, dopo che il computer ha calcolato la prima percentuale, tornasse indietro all'inizio del suo calcolo e ci chiedesse il punteggio successivo. E' possibile farlo usando l'istruzione:

GOTO numero di linee

che ridirige il programma a qualsiasi numero di linea voluto. Abbiamo riscritto il programma di calcolo percentuale in questo modo:

```
10 REM **PERCENTUALE DEL PUNTEGGIO**
15 PRINT CHR$(147)
20 PRINT"CALCOLO PERCENTUALE"
30 PRINT"INSERISCI IL PUNTEGGIO MASSIMO"
40 INPUT T
45 REM **INIZIO INSERIMENTO**
50 PRINT"INSERISCI IL VALORE SUCCESSIVO"
60 INPUT M
70 LET P=(M/T)*100
80 PRINT"PERCENTUALE",P
90 GOTO 45
```

Programma 2 – Programma di calcolo percentuale per uso ripetuto.

Nota le nuove linee 20 e 30, che ci permettono di immettere il valore del punteggio massimo al test, una volta sola. Il calcolo viene effettuato nelle linee da 50 a 70 e, nel leggere la linea 80, il programma torna alla linea 40 per chiederci un altro punteggio.
L'esecuzione tipica sarebbe:

```
CALCOLO PERCENTUALE
INSERISCI IL PUNTEGGIO MASSIMO
? 80
INSERISCI IL VALORE SUCCESSIVO
? 42
PERCENTUALE          52.5
INSERISCI IL VALORE SUCCESSIVO
? 67
PERCENTUALE          83.75
INSERISCI IL VALORE SUCCESSIVO
? 19
PERCENTUALE          23.75
```

```
INSERISCI IL VALORE SUCCESSIVO
? 55
PERCENTUALE          68.75
INSERISCI IL VALORE SUCCESSIVO
?
```

L'istruzione GOTO interrompe la normale esecuzione del programma (nell'ordine di numero di linea). Appena il programma legge GOTO, incondizionatamente trasferisce il controllo al numero di linea indicato nell'istruzione. GOTO è chiamata talvolta "salto incondizionato".

- ESEGUI IL PROGRAMMA 2 -

Premi il tasto RUN STOP quando ti stanchi.

DAV 2

Il programma che segue calcola il quadrato dei numeri (moltiplica cioè il numero per se stesso). Aggiungi una linea GOTO per usare il programma più volte e calcolare il quadrato di numeri successivi.

```
5 REM **CALCOLO QUADRATI**
10 INPUT N
20 LET S=N*N
30 PRINT S
40 END
```

Programma 3 – Calcolo dei quadrati.

- Esegui e fai girare le linee da 10 a 40 - Aggiungi le tue linee extra e fai girare il nuovo programma.

2.4 Lo stile di programmazione

L'istruzione GOTO ci ha aiutato in molti modi. Comunque, lascia ancora indefinito qualche aspetto, come la presenza del punto interrogativo {?}, alla fine dell'esecuzione.
Raggiunta la linea 80 nel programma 2, il controllo torna sempre alla linea 40 che poi genera la richiesta di un input ulteriore; da qui il {?}. Il programma in questo modo è inserito in un ciclo perpetuo da cui non può uscire.

Il solo modo per fermarlo è bloccarlo, premendo il tasto di controllo insieme con un altro tasto (il modo di fermare un programma differisce per ogni sistema e dovrai controllare come si fa sul micro che stai usando). Usa i tasti RUN STOP sul computer Commodore 64.

Interrompere l'esecuzione del programma a metà comporta uno stile di programmazione poco efficace, cercheremo pertanto di trovare una soluzione migliore. Dovevamo avvertirti comunque dei pericoli dell'uso del GOTO. Come abbiamo detto, quest'istruzione ti permette virtualmente di saltare ad una qualsiasi posizione nel programma, cosa che, a questo stadio, può sembrare molto utile. Ma poiché GOTO ci permette di saltare a caso ad un punto qualsiasi del programma, spesso è usato in modo tale che la struttura della soluzione, piuttosto che seguire la struttura logica dell'analisi del problema, sia spezzata da "salti di convenienza" ad altre parti di programma. Staremo perciò molto attenti ad usare l'istruzione GOTO in questo corso. La useremo solo quando pensiamo che, non usandola, la chiarezza del programma venga alterata.
Speriamo che anche tu proverai e seguirai il nostro esempio e userai l'istruzione GOTO il meno possibile. Ne eviteremo l'uso indiscriminato, cosa che consigliamo spesso anche sui libri di testo e riviste specializzate.

Noterai che useremo quasi sempre GOTO per tornare alla linea REM. In questo modo, potremo dare una spiegazione del salto nell'istruzione REM ed evidenziare il salto, nella logica di programmare, ci faciliterà la visione del listing. Sarà inoltre più agevole programmare semplici listing che ci permetteranno di aggiungere o rimuovere delle linee senza cambiare i dettagli dell'esecuzione del GOTO.

2.5 IF...THEN...

Il programma che abbiamo appena trattato è: come segnalare al computer che abbiamo raggiunto la fine della lista dei valori? Quando si fa un calcolo manuale possiamo vedere direttamente che abbiamo raggiunto la fine.
Tra breve impareremo come usare il computer per contare in nostra vece, ma prima introdurremo un elemento per segnalare che è stata raggiunta la fine della lista.

Un metodo possibile è il seguente: poni fine alla lista di numeri con un numero speciale che è impossibile ottenere altrimenti, es. –9999: si tratta di un

valore eclatante, non ci aspetteremmo mai che uno studente ottenga il punteggio –9999 in un test! Questo valore è chiamato "impossibile".
Vogliamo che il programma giri normalmente quando vengono immessi valori ma che si fermi quando viene immesso il valore –9999.
In altre parole, vogliamo poter scrivere un programma con la seguente struttura logica:

1. Inizio
2. Immetti il valore del punteggio massimo
3. Immetti il punteggio successivo
4. Se questo punteggio è uguale a –9999, vai al punto 8 altrimenti vai al punto 5
5. Calcola la percentuale
6. Da come risultato la percentuale
7. Vai al punto 3
8. Stop

Figura 2 – Come porre fine al calcolo della percentuale.

Fortunatamente esiste un'istruzione BASIC con cui è possibile prendere una decisione alla linea 4:

```
IF.....THEN numero di linea
  └─condizione da soddisfare per saltare ad un dato numero di linea.
```

Perciò tutto quello che bisogna fare è tradurre l'istruzione 4 della figura 2, in

```
65 IF M = -9999 THEN 100
```

e inserirla nel programma 4. Quest'istruzione significa: se il valore trovato in M è uguale a –9999, allora vai alla linea 100, altrimenti continua eseguendo l'istruzione successiva a 65.
L'istruzione della Figura 2 "altrimenti vai al punto 5" non è tradotta in BASIC, ma è implicita: o esci fuori dalla sequenza o continui in sequenza.
Possiamo farlo agevolmente usando alcuni nuovi numeri di linea da porre tra quelli già utilizzati. (Ricorderai che deliberatamente abbiamo etichettato le istruzioni 10, 20, 30... in modo da lasciare spazio per altre possibili istruzioni successive). Avremo:

```
10 REM **PERCENTUALI**
15 PRINT CHR$(147)
20 PRINT"CALCOLO PERCENTUALE"
30 PRINT"INSERISCI IL PUNTEGGIO MASSIMO"
40 INPUT T
45 REM **INIZIO INSERIMENTO**
50 PRINT"INSERISCI IL VALORE SUCCESSIVO"
60 INPUT M
65 IF M=-9999 THEN 100
70 LET P=(M/T)*100
80 PRINT"PERCENTUALE",P
90 GOTO 45
100 END.
```

Programma 4 – Calcolo dalla percentuale con un valore impossibile.

Usa il programma 4 esattamente come il programma 2 finché vuoi immettere valori da convertire in percentuale. Poi immetti il valore –9999 e il programma ha termine.

Ecco un'esecuzione tipica:

```
RUN
CALCOLO PERCENTUALE
INSERISCI IL PUNTEGGIO MASSIMO
? 80
INSERISCI IL VALORE SUCCESSIVO
? 43
PERCENTUALE          53.75
INSERISCI IL VALORE SUCCESSIVO
? 29
PERCENTUALE          36.25
INSERISCI IL VALORE SUCCESSIVO
? 62
PERCENTUALE          77.5
INSERISCI IL VALORE SUCCESSIVO
? -9999 ———————————————— Valore per terminare

READY.
```

- ESEGUI IL PROGRAMMA 4 - usalo per convertire altri valori. Termina l'esecuzione con –9999.

DAV3

La DAV2 richiedeva che tu scrivessi il programma:

```
10 INPUT N
20 LET S=N*N
30 PRINT S
35 GOTO 10
40 END
```

ma questo, come il programma di percentuale, non si ferma mai. Modificalo includendo un valore terminale che ponga fine all'esecuzione.

2.6 Le disuguaglianze

Sarebbe utile poter usare l'espressione M=–9999 nell'istruzione IF...THEN.... per determinare se ha luogo o meno una diramazione all'interno del programma. L'istruzione significa: se M=–9999 è vera, allora vai alla diramazione, altrimenti continua. Il segno = stabilisce una relazione tra M e –9999.

Il BASIC permette di usare espressioni che includono relazioni:

Relazioni	**Esempio**	**Significato**
>	A > B	il valore nella locazione di memoria A è maggiore del valore in B
<	X < Y	il valore nella locazione di memoria X è minore del valore in Y

Vero o falso?

Considera l'espressione A < B. Se A=2 e B=5 allora A < B è vero, perché 2 è minore di 5. Considera la stessa espressione con i valori A=2 e B=1. Ora A < B è falso perché 2 non è minore di 1. E così, se A=2 e B=2 allora A < B è falso perhé 2 non è minore di 2. Nello scrivere programmi, spesso ci sarà utile poter sapere se un'istruzione che include i simboli = o < o > sarà vera o falsa. Parliamo di errato logico dell'asserzione, se per es.:

Asserzione	**Stato logico**
3 > 2	vero
7 < 7	falso

Probabilmente tutto questo ti sembrerà abbastanza facile per i numeri interi positivi, ma puoi essere meno sicuro di ciò che accade negli altri casi.
Se hai dubbi, ricorda la rappresentazione dei valori numerici su una retta

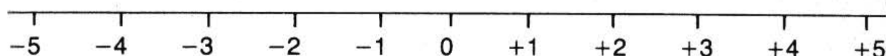


Figura 3 – Rappresentazione dei valori numerici su una retta.

Se un numero A si trova a sinistra di un secondo numero B, allora A è minore di B, se si trova a destra allora A è più grande.

Esempio 1

Controlla se le seguenti espressioni sono vere o false per valori dati di A e B.

Valori		Espressione
A	B	
2	5	A>B
2	−5	A>B
−2	−5	A>B
−2	−1	A>B
−5	2	A<B
5	−2	A<B
−3	3	A=B

Soluzione

Ricaviamo lo stato logico di ogni espressione usando i valori dati e, aiutati dalla retta numerata, decidiamo se l'asserzione è vera o falsa per quei valori particolari.

Perciò la soluzione è

Valori		Asserzione		
A	B	Espressione	Valore	Stato logico
2	5	A>B	2> 5	F
2	−5	A>B	2>−5	T
−2	−5	A>B	−2>−5	T
−2	−1	A>B	−2>−1	F
−5	2	A<B	−5< 2	T
5	−2	A<B	5<−2	F
−3	3	A=B	−3= 3	F

dove F = falso e V = vero

DAV4

Completa la tabella seguente per determinare se le espressioni sono vere o false per i valori dati.

Valori		Asserzione		
A	B	Espressione	Valore	Stato logico
3	7	A>B		
5	3	A>B		
−3	5	A>B		
8	5	A<B		
3	9	A<B		
8	−2	A<B		

Ora siamo pronti ad usare relazioni per fare in modo che il controllo di un programma salti ad una nuova linea, quando certe condizioni vengono soddisfatte.

Esempio 2

Nel seguente segmento di programma, dopo l'esecuzione della linea 30, il controllo passerà alla linea 40 o alla linea 100?

```
10 LET A=-3
20 LET B=2
30 IF A+B>0 THEN 100
40 REM
```

Programma 5

Soluzione

–3 + 2 > 0 è falso, perciò la diramazione a 100 non avverrà e il controllo passerà alla linea 40.

DAV5

Nei seguenti segmenti di programma, dopo l'esecuzione della linea 30, il controllo passerà alla linea 40 o alla linea 100?

(a)
```
10 LET A=7
20 LET B=-8
30 IF A-B<0 THEN 100
40 REM
```

(b)
```
10 LET X=3
20 LET Y=-3
30 IF X/Y=-1 THEN 100
40 REM
```

(c)
```
10 LET P=-1
20 LET Q=3
30 IF P+Q>Q THEN 100
40 REM
```

(d)
```
10 LET M=3
15 LET N=-4
20 LET P=-2
30 IF M-N<N-P THEN 100
40 REM
```

(e)
```
10 LET R=1
20 LET S=-2
30 IF R+S>-1 THEN 100
40 REM
```

Programmi 6–10

2.7 I diagrammi di flusso

Come abbiamo detto, il compito principale di un programmatore è quello di trovare un modo idoneo di esprimere la strategia per risolvere un problema particolare. A questo punto, dobbiamo introdurre un concetto fondamentale nel gergo del computer: algoritmo.

Questo termine viene usato per indicare una strategia di soluzione generale ed è definito come una serie di istruzioni o passi procedurali per la soluzione di

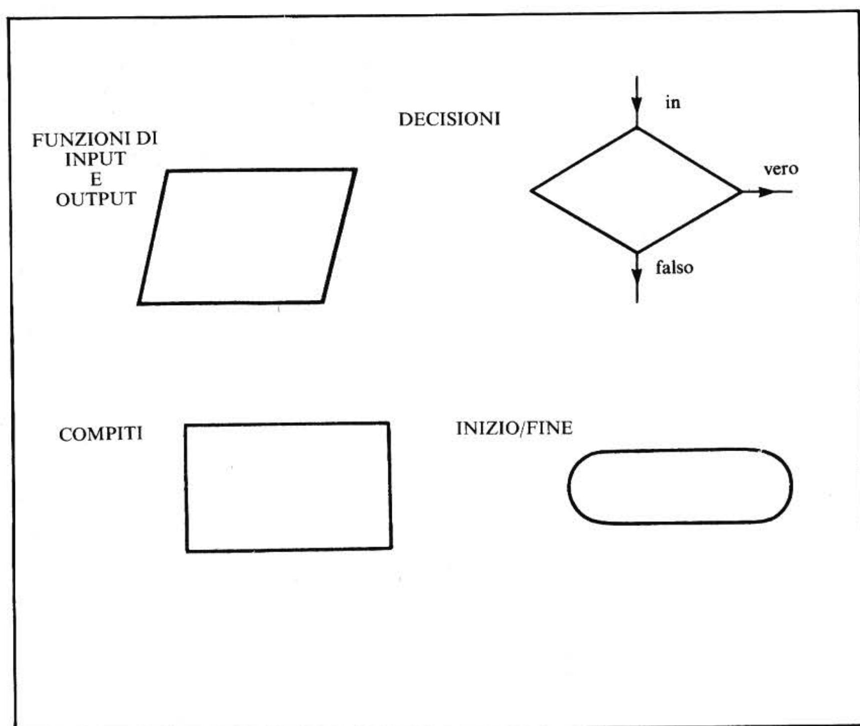
un problema specifico. Noterai che in questo caso il computer non è stato menzionato.
A parte questo, la definizione di programma e algoritmo è identica. Un programma , in definitiva, è un algoritmo scritto per un computer.

Ci sono tre modi basilari per definire un algoritmo:

(i) una descrizione
(ii) la codifica in BASIC
(iii) un diagramma di flusso o diagramma a blocchi.

I diagrammi di flusso sono evidenziati dal colore e si rivolgono a chi di noi ama vedere gli eventi rappresentati in modo pittorico.

Illustriamo le differenti funzioni all'interno di un algoritmo usando "blocchi" geometrici differenti.



Il primo programma dell'unità didattica 1 può essere espresso in forma di diagramma di flusso come segue:

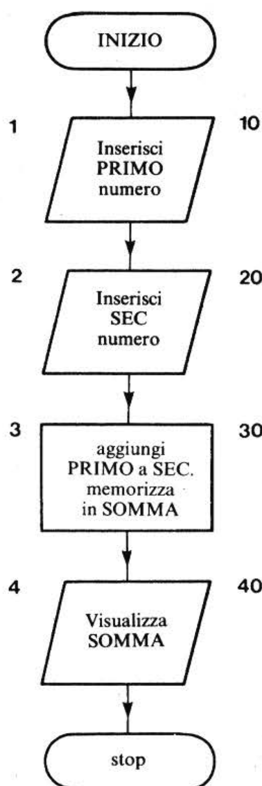


Figura 4 – Diagramma di flusso del programma 1 dell'unità

Le descrizioni delle funzioni nei blocchi nonostante siano in forma abbreviata, potrebbero essere eseguite anche da chi non abbia conoscenza del BASIC. Perciò cercheremo di mantenere indipendenti dal linguaggio queste descrizioni. I numeri a sinistra dei blocchi si riferiscono alle istruzioni nell'algoritmo descritto in figura 4 dell'unità didattica 1; i numeri a destra si riferiscono alle istruzioni BASIC nel programma 1 dell'unità didattica 1.

DAV6

Costruisci un diagramma di flusso per il programma di percentuale (programma1).

Il blocco di decisione

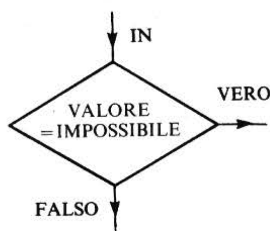
Hai già visto come vengono effettuate le decisioni in BASIC usando l'istruzione IF...THEN... La logica è:

IF (se) l'asserzione è vera, THEN (allora) vai alla linea X altrimenti (se l'asserzione è falsa) vai alla linea successiva.

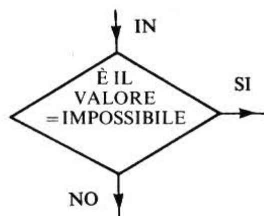
L'idea base della diramazione alla linea X o del continuare alla linea successiva è illustrata, nel diagramma di flusso, da 2 linee in uscita da un blocco di decisione. La decisione della linea 40 del programma.

65 IF M = -9999 THEN 100

potrebbe essere illustrata in una forma indipendente dal linguaggio:



L'asserzione VALORE = IMPOSSIBILE potrebbe essere espressa sotto forma di domanda:



Lo stile del diagramma di flusso non dovrebbe risultarti troppo complesso. Il controllo dell'efficacia di un diagramma di flusso consiste nel fatto che tu possa seguirlo più o meno facilmente qualche tempo dopo la sua formulazione. Oppure, se stai cercando di comunicare le tue idee a qualcun altro, questi potrà seguire con facilità il tuo diagramma di flusso.

Qui sotto viene riportato un diagramma di flusso relativo al programma 4.

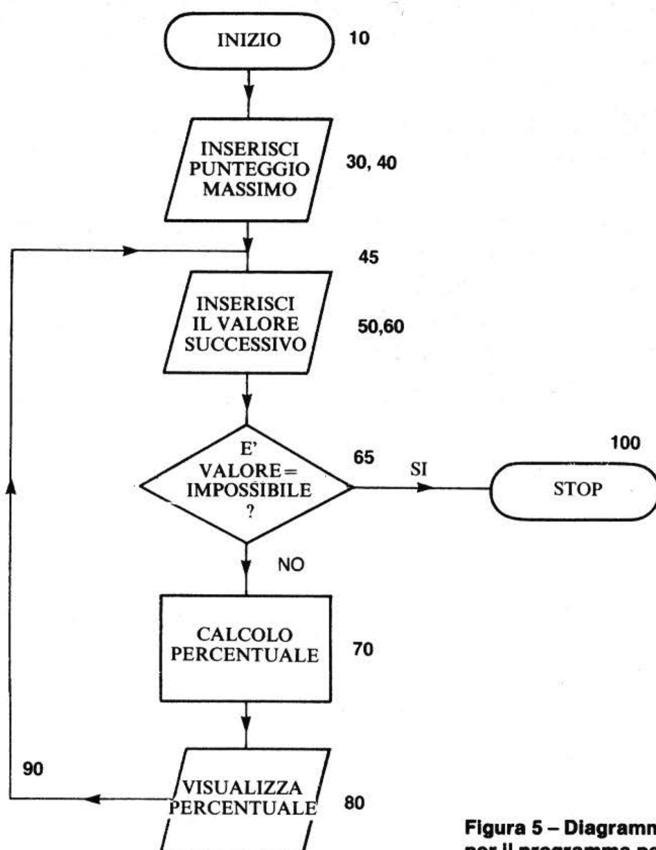


Figura 5 – Diagramma di flusso per il programma percentuale.

I numeri a destra dei blocchi del diagramma di flusso corrispondono ai numeri di etichetta del programma.
L'istruzione 90 GOTO 50 è rappresentata da una freccia all'indietro verso il blocco 50.

DAV7

Scrivi un diagramma di flusso per il programma che hai scritto in risposta alla DAV3.

Ora che abbiamo introdotto l'idea di diagrammi di flusso, possiamo usarli per pianificare le strutture dei programmi che scriveremo in futuro.

2.8 Contare

Come abbiamo detto, i computer sono bravi ad eseguire numerose procedure ripetitive. Se comunque desideriamo controllare queste attività, piuttosto che avviarle e fermarle solamente, come abbiamo fatto nell'ultimo esempio, allora dobbiamo fare in modo che il computer conti per noi le ripetizioni. Se vogliamo ripetere un'attività un numero determinato di volte, cominciamo col numerare la prima attività, aggiungiamo uno per ogni attività successiva, fino a che raggiungiamo un limite predeterminato. Possiamo illustrare questa procedura in forma di diagramma di flusso.

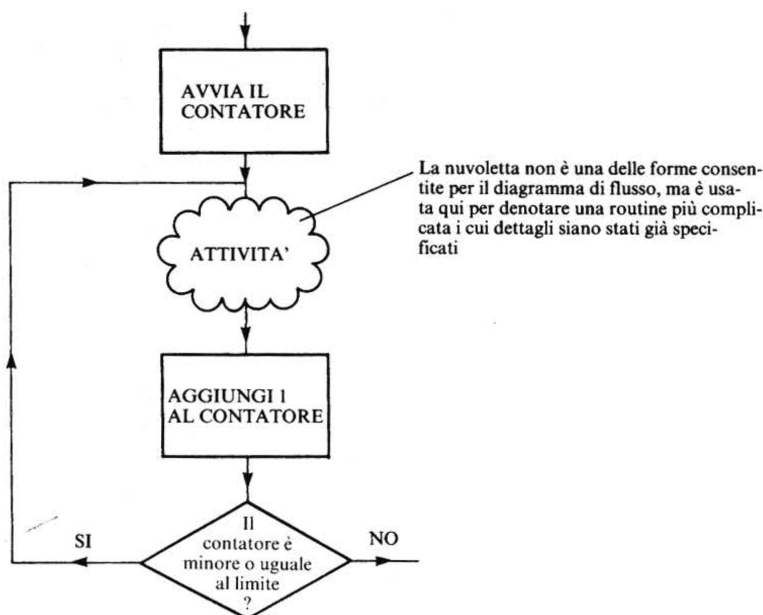


Figura 6 – Un contatore nel diagramma di flusso.

Nota che ci sono tre parti nel contatore:

(i) la procedura che pone il contatore al valore iniziale

(ii) la procedura che aggiunge 1 al contatore ogni volta che l'attività è completata
(iii) la procedura per fermare il contatore e lasciare l'attività quando questa sia stata eseguita il numero richiesto di volte.

Dobbiamo prestare molta attenzione, per essere sicuri di uscire da un tale ciclo ripetitivo esattamente al punto che vogliamo. Attento, nota che se, per es., il ciclo ha contato fino a 10, il valore della locazione COUNT, nel lasciare il ciclo, sarà 11. Questo è un punto a cui dovrai stare molto attento se desideri usare il numero in COUNT, più avanti, nel programma.

DAV8

Quanti numeri saranno immessi con i diagrammi di flusso seguenti?

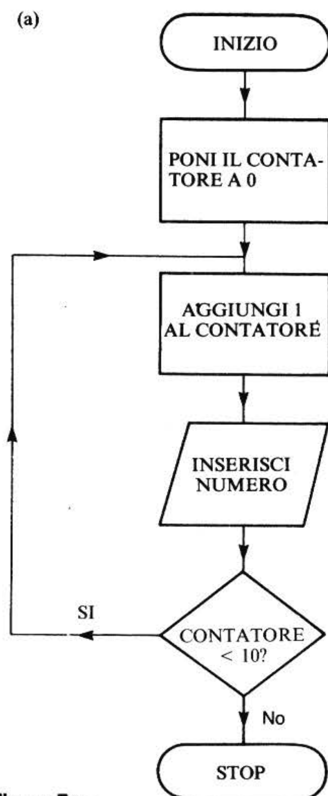


Figura 7a –

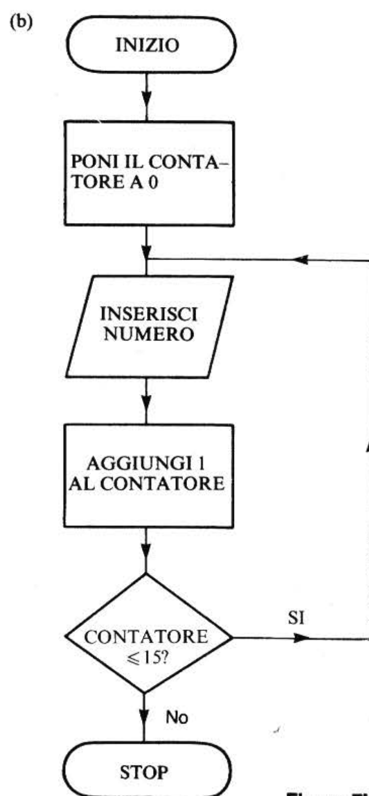


Figura 7b –

DAV9

Fà in modo che il seguente programma legga 5 numeri completando l'istruzione IF...THEN...

```
7 PRINT CHR$(147)
10 LET C=0
15 REM **LEGGI IL PROSSIMO NUMERO**
20 INPUT N
30 IF C=4 THEN 55
40 LET C=C+1
50 GOTO 15
55 REM ** FATTO**
60 END
```

Programma 11

Esempio 3

Scrivi un programma BASIC per calcolare e stampare le percentuali dei punteggi relative ad un test di un gruppo di 5 studenti.

Soluzione

Se assumiamo che l'"attività" nella nuvoletta del diagramma di flusso della figura 6 sia

immetti il punteggio calcola la percentuale stampa la percentuale

allora possiamo illustrare l'algoritmo di questa soluzione in forma di diagramma di flusso.

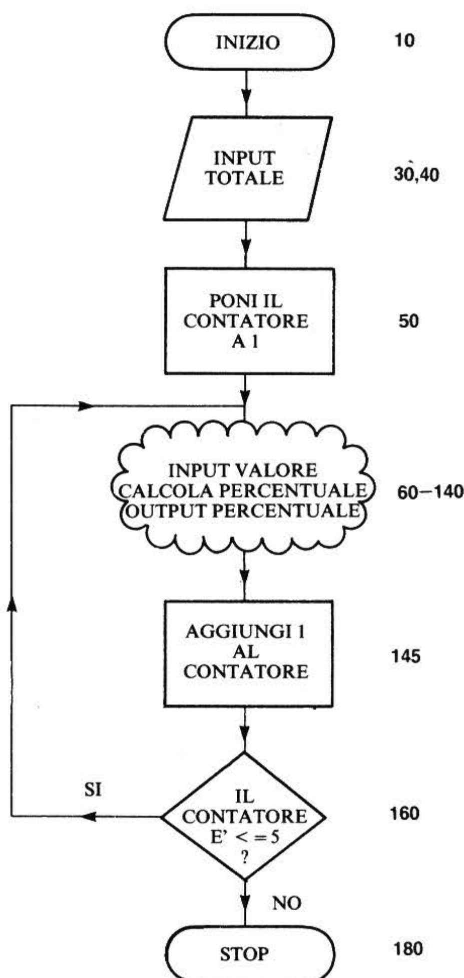


Figura 8 – Il diagramma di flusso per il calcolo della percentuale su 5 valori.

Questo diagramma ci illustra la struttura del programma che dobbiamo scrivere: può essere riscritto modificando il programma 4 per incorporare il

contatore. Perciò, il programma richiesto è:

```
10 REM **PERCENTUALE**
20 REM **PUNTEGGIO**
25 PRINT CHR$(147)
30 PRINT"INSERISCI IL PUNTEGGIO MASSIMO"
40 INPUT T
50 LET C=1 ——————— Parte il contatore, "Inizializzazione"
60 REM **NUOVO CICLO**
70 PRINT"INSERISCI IL VALORE SUCCESSIVO"
80 INPUT M
120 REM **CALCOLO**
130 LET P=(M/T)*100                Aggiunge 1 al calcolatore
140 PRINT"PERCENTUALE",P          Incremento
145 LET C=C+1 ————————————————————/
150 PRINT"LINEA 150,"              Questa linea per provare quello che
152 PRINT"IL CONTATORE E'=",C ——— sta succedendo al contatore a
160 IF C<=5 THEN 60                questo punto del programma
170 REM **FATTO**       \Conteggio
180 END                  completato
```

Programma 12 – Aggiunta di un contatore al programma di calcolo delle percentuali.

```
INSERISCI IL PUNTEGGIO MASSIMO
? 75
INSERISCI IL VALORE SUCCESSIVO
? 57
PERCENTUALE             76
LINEA 150,IL CONTATORE E'=      2
INSERISCI IL VALORE SUCCESSIVO
? 62
PERCENTUALE             82.6666667
LINEA 150,IL CONTATORE E'=      3
INSERISCI IL VALORE SUCCESSIVO
? 43
PERCENTUALE             57.3333333
LINEA 150,IL CONTATORE E'=      4
INSERISCI IL VALORE SUCCESSIVO
? 39
PERCENTUALE             52
LINEA 150,IL CONTATORE E'=      5
```

```
INSERISCI IL VALORE SUCCESSIVO
? 70
PERCENTUALE              93.3333334
LINEA 150,IL CONTATORE E'=        6

READY.
```

Il programma è preparato per interrompersi quando il contatore arriva a 6.

- ESEGUI PROGRAMMA 12 -

Esercizio 1

In risposta alla domanda 2 del compito 1, dovresti aver scritto un programma per calcolare il costo dell'installazione di un doppio vetro in una finestra. Se volessi usare un programma per calcolare i costi di più finestre, dovresti far girare il programma più volte. Questo esercizio ti chiede di modificare il programma in questo senso.
Nota che l'"attività" di ripetizione sarà

Immetti base e altezza della finestra calcola il costo di installazione della finestra visualizza il costo

(a) Traccia un algoritmo in forma di diagramma di flusso per calcolare ed visualizzare i costi di sei finestre
(b) codifica l'algoritmo in BASIC. - esegui il nuovo programma -, prova a farlo girare sul tuo microcomputer, per le sei finestre
(c) estendi il tuo programma a qualsiasi numero di finestre, da specificare all'inizio.

Esercizio 2

Estendi il problema posto nella domanda del compito 1.

L'attività da ripetere sarà:

calcola il ricavo visualizza l'anno e il ricavo relativo calcola il deposito per l'anno successivo

(a) Traccia un algoritmo in forma di diagramma di flusso per calcolare ed visualizzare il ricavo per ogni anno fino a sei anni. Controlla la tua risposta.
(b) Codifica questo algoritmo in BASIC, in dettaglio.
Controlla la tua risposta, - esegui il programma -.
c) Estendi l'algoritmo per calcolare ed emettere il ricavo per ogni anno fino ad un numero stabilito di anni, da specificare all'inizio del programma.
Controlla la tua risposta, - esegui il programma -.

2.9 Confronti

Abbiamo visto come il linguaggio BASIC ci permette di confrontare due numeri. Spesso, vogliamo decidere se un valore particolare è più grande o più piccolo di un altro. Questo è un processo fondamentale per ordinare dati. Nel corso, considereremo in maggiore dettaglio i metodi di ordinamento dei dati. Incominciamo, perciò, con il caso più semplice.

Esempio 4

Formula un algoritmo in forma descrittiva per immettere 2 numeri ed ottenere il più grande dei due.

Commento

In questa unità didattica, abbiamo espresso buona parte dei nostri algoritmi sotto forma di diagramma di flusso; questa volta invece, useremo il metodo descrittivo introdotto nell'unità didattica 1.

Soluzione

1. Inizio
2. Inserisci il primo numero
3. Inserisci il secondo numero
4. Se il primo numero > del secondo allora vai a 7, altrimenti vai a 5
5. Visualizza il secondo numero
6. Vai a 8
7. Visualizza il primo numero
8. Stop

Questa non è la soluzione più chiara ma è la più vicina al "primo tentativo" di chi legge il problema. Cercheremo soluzioni più chiare quando torneremo al metodi di ordinamento, nell'unità didattica 9.

Compito 2

1. Formula un diagramma di flusso e scrivi un programma BASIC per inserire due numeri ed ottenere il più piccolo dei due. Modifica il programma in modo che elabori
(a) cinque coppie di numeri
(b) un numero qualsiasi di coppie di numeri

2. Estendi l'algoritmo "valore percentuale" delle pagine precedenti ed e-sprimilo in forma di diagramma di flusso e di programma BASIC
(a) per una classe con un numero qualsiasi di studenti
(b) per calcolare il valore medio percentuale
(c) per calcolare il valore maggiore

Obiettivi dell'unità 2

Ora che hai completato questa unità didattica, controlla se sei capace di:

Combinare stampa letterale e stampa variabile in istruzioni PRINT ☐

Usare, per spaziare, le istruzioni PRINT ☐

Usare GOTO per ripetere l'uso di un programma ☐

Usare un valore impossibile, per porre fine ad un programma ☐

Usare IF...THEN... ☐

Trovare lo stato logico di asserzioni contenenti >, <, = ☐

Costruire diagrammi di flusso ☐

Inserire contatori per controllare l'uso ripetuto di parti di un programma o di un diagramma di flusso ☐

Risposte alle DAV e agli esercizi

DAV1

(a) AREA 48
(b) BASE 8 ALTEZZA 6 AREA 48
(c) BASE ALTEZZA AREA
8 6 48
(d) PRINT "BASE", B, "ALTEZZA", C
(e) PRINT "BASE", B
PRINT "ALTEZZA", C
PRINT "AREA", A
(f) PRINT "BASE", "ALTEZZA", "STAMPA UNO SPAZIO BIANCO", "AREA"

DAV2

Devi solo aggiungere:

35 GOTO 10

DAV3

```
5 REM **CALCOLO QUADRATI**
10 INPUT N
15 IF N=-9999 THEN 40
20 LET S=N*N
30 PRINT S
35 GOTO 5
40 REM **FATTO**
50 END
```

Programma 13

Nota che in questo caso il valore terminale "–9999" non è adatto. Puoi non avere un punteggio –9999, ma potresti forse volere il quadrato di –9999, questo programma si rifiuterebbe di calcolarlo.

DAV4

Valori		Asserzioni		
A	B	Espressioni	Valore	Stato logico
3	7	A>B	3>7	F
5	3	A>B	5>3	T
−3	5	A>B	−3>5	F
8	5	A<B	8<5	F
3	9	A<B	3<9	T
8	−2	A<B	8<−2	F

Se uno di questi valori è sbagliato, guarda di nuovo la retta numerata:

A ——————— B ——— A alla sinistra di B significa A < B

B ——————— A ——— A alla destra di B significa A > B

DAV5

(a) 40 (b) 100 (c) 40
(d) 40 (e) 40

Il tuo computer potrebbe aiutarti a risolvere il problema.

Le istruzioni

40 PRINT "40"

e 100 PRINT "100"

faranno sì che venga emessa la linea giusta.

I programmi seguenti mostrano come potresti avere risolto (a) e (b)

Programma da risolvere (a)

```
10 LET A=7
20 LET B=-8
30 IF A-B<0 THEN 100
40 PRINT"40"
50 GOTO 999
100 PRINT"100"
999 END
```

Programma 14

Programma da risolvere (b)

```
10 LET X=3
20 LET Y=-3
30 IF X/Y=-1 THEN 100
40 PRINT"40"
50 GOTO 999
100 PRINT"100"
999 END
```

Programma 15

Effetto dell'esecuzione del programma 14

```
RUN
40

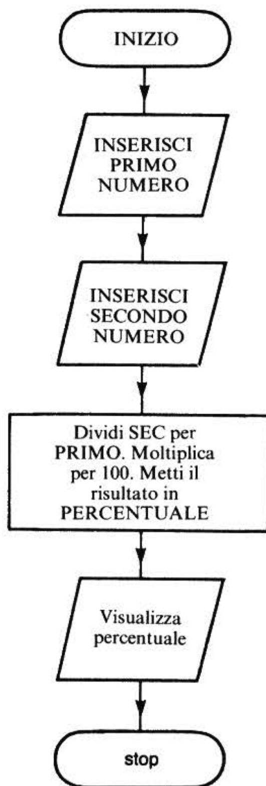
READY.
```

Effetto dell'esecuzione del programma 15

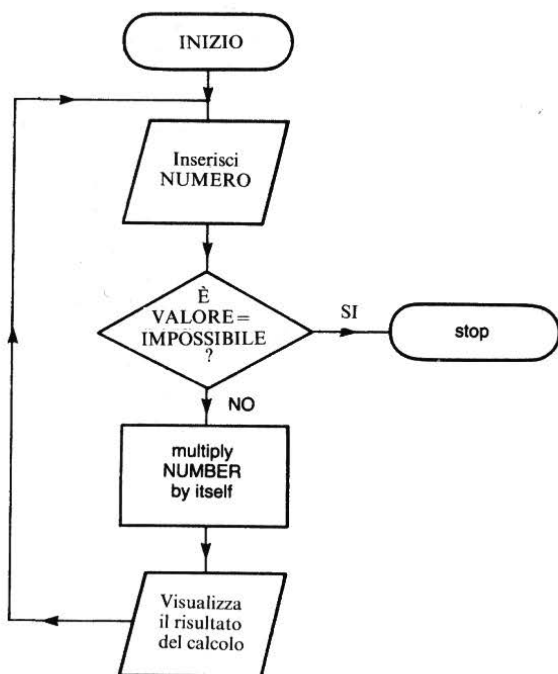
```
RUN
100

READY.
```

DAV6



DAV7



DAV8

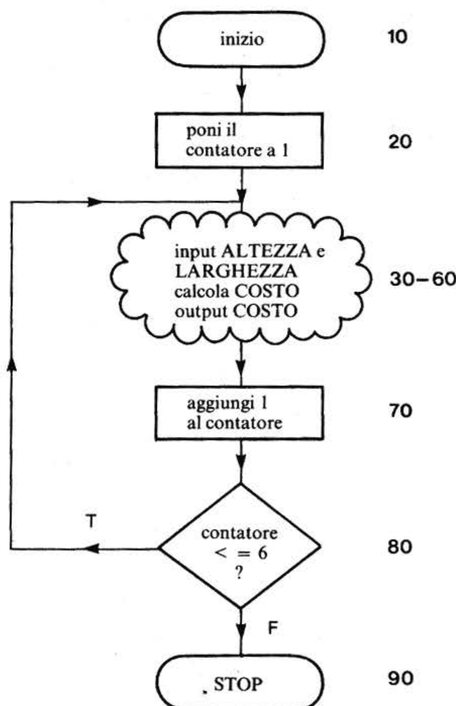
(a) 10; (b) 16

DAV9

```
30 IF C = 5 THEN 60
```

Esercizio 1

1 (a)



1 (b)

```
10 REM **FINESTRA DOPPPIO VETRO**
15 PRINT CHR$(147)
20 LET C=1
25 REM **PROSSIMO INSERIMENTO**
30 PRINT"INSERISCI L'ALTEZZA IN METRI"
35 INPUT H
40 PRINT"INSERISCI LA LARGHEZZA IN METRI"
45 INPUT W
50 LET K=14000*(H+W)+40000*H*W+50000
60 PRINT"FINESTRA",C,"COSTO",K
70 LET C=C+1
80 IF C<=6 THEN 25
90 END
```

Le linee 20, 70 e 80 rappresentano il contatore

Programma 16

```
RUN

INSERISCI L'ALTEZZA IN METRI
? 1.5
```

```
INSERISCI LA LARGHEZZA IN METRI
? 2
FINESTRA   1         COSTO      219000
INSRERISCI L'ALTEZZA IN METRI
? 1.5
INSERISCI LA LARGHEZZA IN METRI
? 3
FINESTRA   2         COSTO      293000
INSERISCI L'ALTEZZA IN METRI
? 1.5
INSERISCI LA LARGHEZZA IN METRI
? 4
FINESTRA   3         COSTO      367000
INSERISCI L'ALTEZZA IN METRI
? 2.5
INSERISCI LA LARGHEZZA IN METRI
? 2
ecc.
```

1(c)

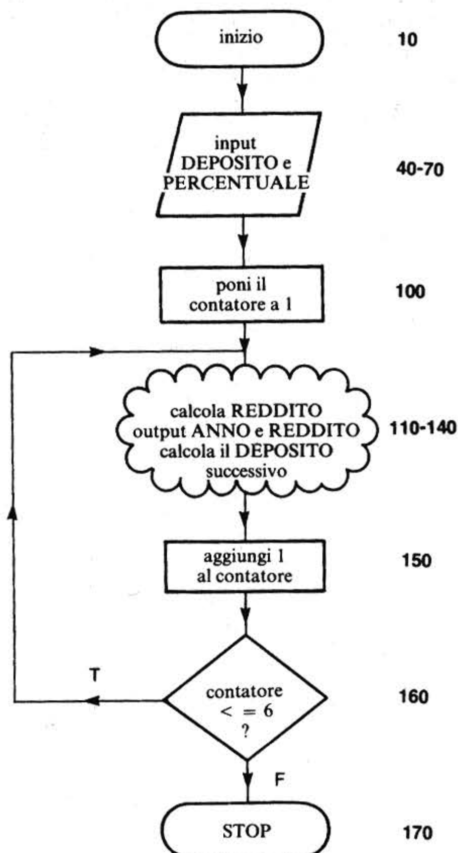
```
10 REM **COSTO DEL DOPPIO VETRO**
12 PRINT"INSERISCI IL NUMERO DELLE FINESTRE"
14 INPUT N
20 LET C=1
30 PRINT"INSERISCI L'ALTEZZA IN METRI"
35 INPUT H
40 PRINT"INSERISCI LARGHEZZA IN METRI"
45 INPUT L
50 LET K=3000*(2*H+2*L)+4000*(2*H+2*L)+4000*H*L+50000
60 PRINT"FINESTRA",C,"COSTA",K
70 LET C=C+1
80 IFC <=N THEN 30
90 END
```

Programma 17

Se scrivi PRINT in un programma, senza farlo seguire da altro, sullo schermo viene stampata una linea vuota. E' questa una soluzione utile per spaziare le linee sullo schermo: blocchi troppo compatti di parole risultano, altrimenti, di difficile lettura.

Esercizio 2

2(a)



2(b) Programma

```
10 REM **INTERESSI COMPOSTI**
15 PRINT CHR$(147)
20 PRINT"TABELLA INTERESSI COMPOSTI"
30 PRINT
40 PRINT"INSERISCI DEPOSITO L.";
50 INPUT D
60 PRINT"INSERISCI LA PERCENTUALE D'INTERESSE(%)";
70 INPUT P
80 PRINT
```

```
90 REM **CALCOLO**
100 LET C=1
110 REM **INIZIO CICLO DI CALCOLO**
120 LET Y=(P*D)/100
130 PRINT"ANNO";C;"---REDDITO---L.";Y
140 LET D=D+Y
150 LET C=C+1
160 IF C<=6 THEN 110
170 END
```

Programma 18

```
RUN
TABELLA INTERESSI COMPOSTI

INSERISCI DEPOSITO L.? 300000
INSERISCI LA PERCENTUALE D'INTERESSE(%)?
 12.5

ANNO 1 ---REDDITO---L. 37500
ANNO 2 ---REDDITO---L. 42187.5
ANNO 3 ---REDDITO---L. 47460.9375
ANNO 4 ---REDDITO---L. 53393.5547
ANNO 5 ---REDDITO---L. 60067.749
ANNO 6 ---REDDITO---L. 67576.2177

READY.
```

2 (c) Programma

```
10 REM **INTERESSI COMPOSTI**
15 PRINT CHR$(147)
20 PRINT"TABELLA INTERESSI COMPOSTI"
30 PRINT
35 PRINT"INSERISCI IL NUMERO DEGLI ANNI";
36 INPUT N
40 PRINT"INSERISCI IL DEPOSITO L.";
50 INPUT D
60 PRINT"INSERISCI LA PERCENTUALE D'INTERESSE(%)";
70 INPUT P
80 PRINT
90 REM **CALCOLO**
100 LET C=1
```

```
110 REM **INIZIO CICLO DI CALCOLO**
120 LET Y=(P*D)/100
130 PRINT"ANNO";C;"--¬REDDITO¬--L.";Y
140 LET D=D+Y
150 LET C=C+1
160 IF C<=N THEN 110
170 END
```

Programma 19

```
RUN
TABELLA INTERESSI COMPOSTI

INSERISCI IL NUMERO DEGLI ANNI? 4
INSERISCI IL DEPOSITO L.? 300000
INSERISCI LA PERCENTUALE D'INTERESSI(%)?
 13.75

ANNO 1 ---REDDITO---L. 41250
ANNO 2 ---REDDITO---L. 46921.875
ANNO 3 ---REDDITO---L. 53373.6328
ANNO 4 ---REDDITO---L. 60712.5073

READY.
```

UNITÀ 3
Stringhe

3.1 Cos'è una stringa?

Le prime due unità didattiche riguardavano l'elaborazione dei numeri. L'uomo comune vede spesso il computer come un "mangiatore di numeri" ma questa non è certamente la sua funzione principale, specialmente in ambiente commerciale.
In questa unità, vedremo come il computer può essere usato per manipolare caratteri, con il linguaggio BASIC.
Per "carattere", intendiamo l'alfabeto in lettere maiuscole, le dieci cifre 0–9, i segni di punteggiatura ed alcuni caratteri speciali, come segue:

@, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, [, \,], ↑, ←,
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, :, ;, <, =, >, ?,spazio, !, ,, #, $, %, &, ., (,), +, *, −, /.

Hai imparato a scrivere programmi che usano numeri (3, 57, –92, ecc.) e variabili (A, X, Z, ecc.).
Il BASIC ci permette anche di immettere nel computer gruppi di caratteri.

Questi gruppi di caratteri vengono detti "stringhe". Alcuni esempi di stringhe sono:

GATTO	(una parola)
MARIO ROSSI	(un nome)
Z91?27	(un insieme di caratteri)
ROMA 375412H	(un numero di targa)

una stringa, cioè, può essere costituita da un insieme qualsiasi di caratteri; anche uno spazio è un carattere molto importante in una stringa!

Per ciò che riguarda il BASIC un numero viene trattato come tale quando è usato per eseguire qualche operazione aritmetica, altrimenti è considerato una stringa di caratteri numerici.
Quando vediamo un numero di targa o di telefono, lo percepiamo come gruppo di caratteri numerici, perciò non useremmo questo insieme di cifre per fare dei calcoli.

Locazione di memoria per le stringhe

Come possiamo segnalare al computer che i caratteri immessi devono essere trattati come numeri per scopi aritmetici o piuttosto come stringhe di caratte-

ri? In BASIC la distinzione viene fatta dal modo in cui etichettiamo le locazioni di memoria e dal modo in cui introduciamo i caratteri. Se un nome di locazione di memoria viene seguito dal simbolo $, i caratteri immessi in quella locazione sono trattati come stringhe di caratteri.

Ricorderai dall'unità didattica 1 che in un sistema minimale BASIC ci sono 286 locazioni di memoria per i numeri:

A, A0,...A9
B, B0, ecc.

Per le stringhe, in un BASIC minimale, ci sono altre 286 locazioni di memoria:

A$, A0$, ... A9$
etc.
... Z7$, Z8$, Z9$.

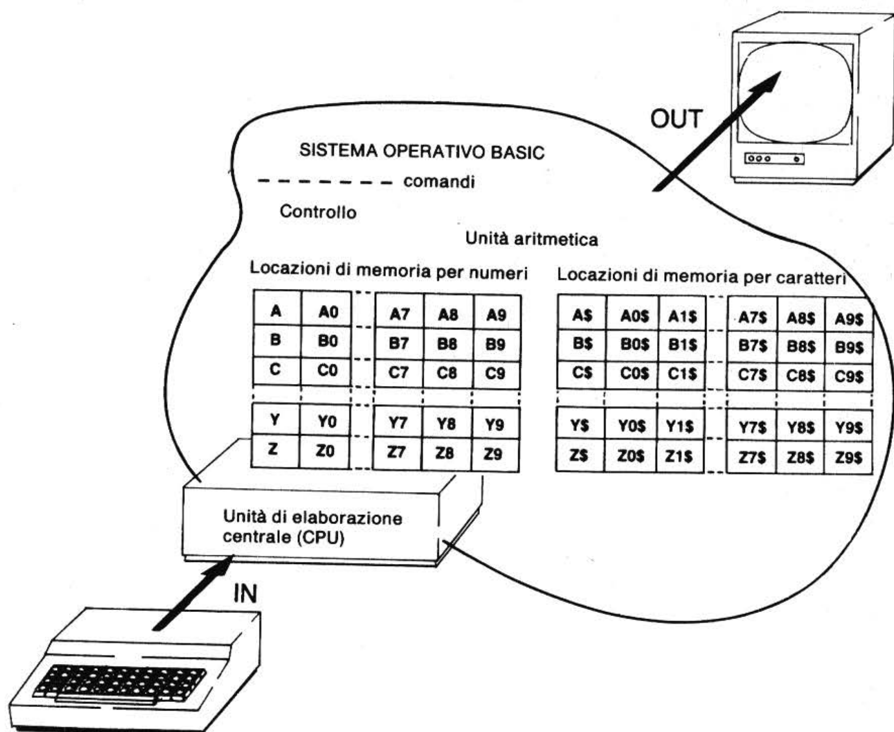


Figura 1 - Dentro il computer.

che si leggono come segue:

A$ A stringa o A dollaro
B9$ B nove stringa o B nove dollaro

Perciò, ora puoi pensare che un microcomputer ha due aree per le locazioni di memoria: una per i numeri e una per i caratteri (vedi Figura 1).

" " con stringhe

Normalmente, dobbiamo dire al computer che la nostra stringa di caratteri va trattata come stringa. Per far ciò, racchiudiamo la stringa tra virgolette. Perciò scriviamo:

10 LET A$="CIAO"
e
30 IF A$="CIAO" THEN 80
e così via

DAV1

Quali dei seguenti sono nomi di locazione di memoria validi per le stringhe?
(a)A$ (b)M8 (c)T7$ (d)B9 (e)C$3 (f)8P$ (g)2$ (h)5$

DAV2

Quali delle seguenti sono istruzioni BASIC corrette?
(a) LET A = 87
(b) LET B$ = "MARIO"
(c) LET M$ = 9583
(d) LET K8 = "LUCA ROSSI"
(e) LET L17 = 38

3.2 Le stringhe

A questo punto, sarebbe pertinente porsi la domanda: "quanto è lunga una stringa?". In altre parole, quanti caratteri possono essere immessi, immagazzinati ed emessi come singolo gruppo?

Ancora una volta, questo dipende dal computer che stai usando. Il Commodore 64 permette di utilizzare stringhe di 255 caratteri di lunghezza.
Inizialmente, assumeremo che una locazione di memoria per le stringhe contenga fino a 40 caratteri e che questo limite riguarderà sia le stringhe in input che quelle in output. Abbiamo pensato di adottare questo limite dal momento che la maggior parte degli schermi ha una larghezza di circa 40 caratteri. A questo punto puoi immaginare che la locazione di memoria per stringa non sia un'unica celletta etichettata, come nell'esempio dell'alveare, ma che abbia 40 sottodivisioni (vedi Fig. 2).

	1	2	3	4	5					10					15					20					25					30					35					40
A$	Q	U	E	S	T	A		F	I	G	U	R	A		R	A	P	P	R	E	S	E	N	T	A															
B$	L	A		L	O	C	A	Z	I	O	N	E		D	I		M	E	M	O	R	I	A																	
C$	D	I		U	N	A		V	A	R	I	A	B	I	L	E		S	T	R	I	N	G	A																
D$																																								
Y$																																								
Z$																																								

Figura 2

Abbiamo parlato delle stringhe come se fossero nuovi concetti, in realtà le hai già incontrate nell'unità didattica 1, dove abbiamo usato l'istruzione PRINT per ottenere la stampa di messaggi racchiusi tra virgolette: la stringa racchiusa tra virgolette veniva emessa letteralmente, carattere per carattere.
Nell'unità 2 abbiamo visto come le virgole tra gli elementi di un'istruzione PRINT provocano la spaziatura delle stringhe sullo schermo o sulla stampante.

3.3 PRINT ...;...

Da quanto detto, capirai che la disposizione dell'informazione sullo schermo è molto importante. Questo vale tanto per le stringhe quanto per i numeri.
Quando manipoliamo un'informazione testuale, per es. stringhe di caratteri in forma di parole o di codici, vogliamo che le stringhe vengano stampate di seguito, come in una frase e non spaziate sullo schermo in zone di stampa. Usiamo allora l'istruzione PRINT...; ...
PRINT H$;T$ prenderà i caratteri dalla locazione di memoria H$ e comincerà a stamparli dalla sinistra dello schermo o della stampante, facendoli seguire immediatamente dai caratteri della locazione T$.

Nelle prossime pagine, simuleremo un servizio di registrazione di dati di un futuro non troppo lontano e lo useremo per comprendere il significato delle stringhe di input e di output.
Cominciamo con lo scrivere un programma che simula un servizio di risposta telefonica.

```
10 REM **RISPOSTA TELEFONICA**
20 PRINT CHR$(147)
30 PRINT"PRONTO"
40 PRINT"INDICHI IL SUO NUMERO TELEFONICO"
50 INPUT T$
60 PRINT
70 PRINT"PRONTO",T$
80 PRINT
90 PRINT"PRONTO",T$
100 PRINT
110 PRINT"PRONTO",T$
120 PRINT
130 PRINT"PRONTO";T$
140 END
```

Programma 1 – Stampa di stringhe.

Per aiutarti ad analizzare questo programma abbiamo scritto a lato la traccia di una tipica esecuzione. La traccia indica quale linea nel programma genera linea corrispondente nell'output.

```
                                                  Tracce
RUN
PRONTO                                                30
INDICHI IL SUO NUMERO TELEFONICO                      40
? 58632                                               50
                                                      60
PRONTO        58632                                   70
                                                      80
PRONTO        58632                                   90
                                                     100
PRONTO        58632                                  110
                                                     120
PRONTO        58632                                  130
```

Commenti

Traccia 50	INPUT T$ ha generato il punto interrogativo {?}. La nostra risposta era 58632, che il computer tratta come stringa e non come numero (se avessimo scritto INPUT T, allora 58632 sarebbe stato trattato come numero).
Traccia 70	Il PRINT ...;... sulla linea 70 stampa il 5 del numero di telefono alla 11ma posizione di stampa sulla linea di output, laddove
Traccia 90	il PRINT...;... della linea 90 stampa il 5 immediatamente adiacente alla 0 di PRONTO.
Traccia 110	Nessuno dei due modi è soddisfacente, ma le linee 110 e 130 mostrano modi alternativi di introdurre gli spazi
Traccia 130	richiesti, o nella stampa di PRONTO (100) o inserendo lo spazio alla destra della stringa nell'istruzione di output.

- ESEGUI IL PROGRAMMA 1 -

DAV3

Studia questo programma e pensa quale sarà la stampa risultante. Scrivi l'output sulla griglia sotto riportata.

```
7 PRINT CHR$(147)
10 PRINT"IMPAGINAZIONE"
20 LET B$="BASIC"
30 LET C$="BASIC"
40 PRINT
50 PRINT B$,C$
60 PRINT
70 PRINT B$;C$
80 PRINT
90 PRINT B$;"  ";C$
100 END
```

Programma 2

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

DAV4

Scrivi un programma che stampi secondo lo schema seguente:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
B	A	S	I	C						B	A	S	I	C					
B	A	S	I	C	B	A	S	I	C	B	A	S	I	C	B	A	S	I	C
B	A	S	I	C		B	A	S	I	C		B	A	S	I	C			

3.4 INPUT "...";...

Abbiamo già usato PRINT "..." come segnale per un'istruzione di INPUT in parecchi dei nostri programmi. La maggioranza delle versioni BASIC permettono di combinare queste due in un'unica istruzione. Perciò, nel programma 1 potremmo sostituire:

```
40  PRINT "INDICHI IL SUO NUMERO TELEFONICO"
50  INPUT T$
```
con
```
50  INPUT "INDICHI IL SUO NUMERO TELEFONICO";T$
```

L'istruzione INPUT genererà comunque sempre il simbolo "?", perciò nel prossimo programma il segnale "?" diventa una domanda diretta.
Anche nell'ultimo programma abbiamo usato la stringa "PRONTO" molte volte. Nel prossimo, ci risparmieremo la fatica di riscriverla immagazzinandola nella locazione H$ all'inizio del programma.

```
10 REM **RISPOSTA TELEFONICA**
20 PRINT CHR$(147)
30 LET H$="PRONTO"
40 PRINT H$
50 INPUT"QUAL'E' IL SUO NUMERO TELEFONICO";T$
60 PRINT
70 PRINT H$,T$
80 PRINT
90 PRINT H$,T$
100 PRINT
110 PRINT H$;" ";T$
120 END
```

Programma 3

```
RUN
PRONTO
QUAL'E' IL SUO NUMERO TELEFONICO? 58632

PRONTO    58632

PRONTO58632

PRONTO 58632
```

- ESEGUI IL PROGRAMMA 3 -

DAV5

Cosa appare sullo schermo se fai girare questo programma assumendo che il tuo nome sia ROSA ROSSI e che tu abbia 45 anni?

```
5 PRINT CHR$(147)
10 LET T$="GRAZIE"
20 INPUT"COME TI CHIAMI";N$
30 PRINT
40 INPUT"QUANTI ANNI HAI";A$
50 PRINT
60 PRINT T$,N$;A$
```

Programma 4

3.5 Numeri e stringhe nelle istruzioni di stampa

Abbiamo immesso il numero telefonico del programma precedente in una locazione di memoria per le stringhe. Usando una locazione numerica incon-

treremmo dei problemi se il numero fosse troppo lungo o contenesse troppi spazi (es. 01 693 4539). Confrontiamo come il BASIC emette questo dato da locazioni di memoria numerica e di stringa.
Nota: in questo programma abbiamo usato le stringhe di caratteri in S$ per stampare sulla pagina una scala numerica.

```
10 REM **RISPOSTA TELEFONICA**
20 PRINT CHR$(147)
30 LET H$="PRONTO"
35 LET S$="1234567890123456789012345"
40 PRINT H$
50 INPUT"IL SUO NUMERO TELEFONICO";T$
55 INPUT"VUOL RIPETERE PER FAVORE";T
60 PRINT S$
70 PRINT H$,T$
75 PRINT H$,T
80 PRINT S$
90 PRINT H$;T$
95 PRINT H$;T
100 PRINT S$
110 PRINT H$;" ";T$
115 PRINT H$;" ";T
120 END
```

Programma 5 – Stampa di stringhe e numeri.

```
RUN
PRONTO
IL SUO NUMERO TELEFONICO? 58632
VUOL RIPETERE PER FAVORE? 58632
1234567890123456789012345________60
PRONTO    58632
PRONTO     58632_________________75
1234567890123456789012345
PRONTO58632
PRONTO 58632_____________________95
1234567890123456789012345
PRONTO 58632
PRONTO  58632___________________115
```

S$ numera ogni posizione di stampa sulla pagina.
Nota come il numero 5 venga stampato alla 12ma posizione essendo la 11ma riservata al segno del numero in T, e che, se il segno è "+", viene stampato al suo posto uno spazio lo stesso succede alle righe 95 e 115.

```
RUN
PRONTO
IL SUO NUMERO TELEFONICO? -58632
VUOL RIPETERE PER FAVORE? -58632
1234567890123456789012345
PRONTO      -58632
PRONTO      -58632____________________75
1234567890123456789012345
PRONTO-58632
PRONTO-58632______________________95
1234567890123456789012345
PRONTO -58632
PRONTO -58632_____________________115
```

Nota l'effetto quando T$ e T sono entrambi -58632

- ESEGUI IL PROGRAMMA 5 -

DAV6

Scrivi un programma per immettere il tuo nome come stringa e la tua età come numero e per ottenere il messaggio: "Mi chiamo .... ed ho ... anni", con la spaziatura normale.

Il servizio di registrazione dati

Vediamo ora un altro esempio di come, in BASIC, si può disporre una stampa sullo schermo.
Possiamo immaginare che in un futuro non troppo lontano, il nostro televisore, il telefono e il computer saranno collegati insieme come terminale intelligente. Nel leggere un annuncio pubblicitario interessante, possiamo comporre il numero di telefono e potrebbe seguire il seguente dialogo:

	Traccia
RUN	
PRONTO	30
SERVIZIO DI REGISTRAZIONI RICHIESTE	40
	50
FORNISCA LE INFORMAZIONI RICHIESTE	60
	70
NOME? MARIO ROSSI	80
NUMERO TELEFONICO? 833635	90

```
VIA? LORENTEGGIO
NUMERO CIVICO? 77
CITTA'? MILANO
CODICE POSTALE? 20146



GRAZIE

DATI ANAGRAFICI REGISTRATI COME SEGUE:
NOME MARIO ROSSI  TELEFONO 833635
INDIRIZZO: VIA LORENTEGGIO 77
               20146 MILANO

DETTAGLI SUI NOSTRI SERVIZI
LE SARANNO INVIATI AL PIU' PRESTO
I SUOI DATI RIMARRANNO RISERVATI

READY.
```

(Naturalmente invece di "le saranno inviati ..." potrebbe esserci "le forniremo subito, sul terminale, le informazioni che desidera" ed a questo punto sarebbe necessario immettere solo il codice del mittente).

Il nostro dialogo simulato può essere realizzato attraverso il programma seguente:

```
10 REM **REGISTRAZIONE DATI**
20 PRINT CHR$(147)
30 PRINT"PRONTO"
40 PRINT"SERVIZIO DI REGISTRAZIONI RICHIESTE"
50 PRINT
60 PRINT"FORNISCA LE INFORMAZIONI RICHIESTE"
70 PRINT
80 INPUT"NOME";N$
90 INPUT"NUMERO DI TELEFONO";T$
100 INPUT"VIA";R$
110 INPUT"NUMERO CIVICO";H$
120 INPUT"CITTA'";C$
130 INPUT"CODICE POSTALE";P$
140 PRINT
150 PRINT
```

```
160 PRINT
170 PRINT"GRAZIE"
180 PRINT
190 PRINT"DATI ANAGRAFICI REGISTRATI COME SEGUE:"
200 PRINT"NOME ";N$;"  TELEFONO ";T$
210 PRINT"INDIRIZZO: VIA ";R$;" ";H$
220 PRINT"                   ";P$;" ";C$
230 PRINT
240 PRINT"DETTAGLI SUI NOSTRI SERVIZI"
250 PRINT"LE SARANNO INVIATI AL PIU' PRESTO"
260 PRINT"I SUOI DATI RIMARRANNO RISERVATI"
270 END
```

Programma 6 — Registrazione dati.

- ESEGUI IL PROGRAMMA 6 -

3.6 Lettere standard

Un servizio di registrazione dati come quello che abbiamo appena visto, ci sarà in un prossimo futuro, ma lettere standard personalizzate esistono già. Tali lettere possono essere composte su un comune elaboratore di testi, ma se il tuo micro non avesse la possibilità di trattare testi, puoi ugualmente conseguire qualche risultato con il BASIC. Nell'esercizio 2 potrai scrivere una lettera a tuo piacimento.

Esempio 1

Un ufficio di collocamento riceve molte richieste di lavoro. La sua politica è di fare una prima intervista ai candidati idonei, presso l'ufficio locale. Dall'ufficio di collocamento viene inviata ad ogni candidato una lettera standard che contiene i dettagli individualizzati dell'intervista proposta. Pensa ad un programma BASIC che scriva una tale lettera.

Soluzione

Il programma seguente potrebbe farlo:

```
10 REM **LETTERA**
20 PRINT CHR$(147)
30 INPUT A$_____________________nome del candidato
40 INPUT B$_____________________data della lettera del candidato
50 INPUT C$_____________________nome del responsabile
60 INPUT D$_____________________ora del colloquio
70 INPUT E$_____________________data del colloquio
80 INPUT F$_____________________luogo del colloquio
90 INPUT G$_____________________nome del segretario
95 REM **FINE DELL'INSERIMENTO**
100 PRINT
110 PRINT
120 PRINT"EGREGIO ";A$;","
130 PRINT
140 PRINT"GRAZIE PER LA LETTERA DEL"
145 PRINT B$
150 PRINT"LEI PUO' SOSTENERE UN COLLOQUIO CON"
160 PRINT"IL SIGNOR ";C$;" ALLE ORE ";D$
165 PRINT"IL GIORNO ";E$
170 PRINT"NEL NOSTRO UFFICIO DI"
175 PRINT F$
180 PRINT
190 PRINT"DISTINTI SALUTI"
200 PRINT
210 PRINT
220 PRINT
230 PRINT G$
240 END
```

Programma 7 – Lettera di convocazione ad un colloquio

Il risultato dell'esecuzione sarebbe:

```
                                                  Traccia
RUN
? BIANCHI                                              30
? 13 OTTOBRE                                           40
? VERDI                                                50
? 10.00                                                60
? 28 OTTOBRE                                           70
? VIA VENETO 15-ROMA                                   80
? MARIAROSA ROSSI                                      90
                                                      100
                                                      110
EGREGIO BIANCHI,                                      120
                                                      130
GRAZIE PER LA LETTERA DEL                             140
13 OTTOBRE                                            145
LEI PUO' SOSTENERE UN COLLOQUIO CON                   150
IL SIGNOR VERDI ALLE ORE 10.00                        160
IL GIORNO 28 OTTOBRE                                  165
NEL NOSTRO UFFICIO DI                                 170
VIA VENETO 15-ROMA                                    175
                                                      180
DISTINTI SALUTI                                       190
                                                      200
                                                      210
                                                      220
MARIAROSA ROSSI                                       230

READY.
```

L'utente del programma potrebbe trovarlo difficile da usare poiché tutto quello che ottiene è una serie di segnali "?". Potrebbe, allora, preparare uno schema che lo aiuti a ricostruire la struttura della lettera:

```
? A$
? B$
? C$
? D$
? E$
? F$
? G$
```

```
CARO A$,

GRAZIE PER LA LETTERA DEL
B$
LEI PUO' SOSTENERE UN COLLOQUIO CON
C$ ALLE ORE D$
IL GIORNO E$
NEL NOSTRO UFFICIO DI
F$
CORDIALI SALUTI
G$
```

- ESEGUI IL PROGRAMMA 7 -

Esercizio 1

Un agente immobiliare invia periodicamente una lettera per controllare se i clienti del suo archivio stanno ancora cercando casa e se i dati che già possiede (es. tipo di proprietà, il prezzo, etc.) sono corretti. Pensa ad un programma BASIC che scriva una tale lettera.

Esercizio 2

Tutti noi scriviamo lettere che chiedono cose, per esempio dettagli di un prodotto, di un servizio, una vacanza, un lavoro, etc. Pensa ad un programma BASIC per scrivere una lettera che copra la più ampia possibilità di applicazioni, e che lasci a te solo il compito di fornire i dettagli di ogni richiesta.

3.7 File, READ con DATA

READ con DATA

Precedentemente, durante l'esecuzione di un programma, abbiamo immesso dati nella tastiera, in risposta ad una istruzione di input. Un altro modo di introdurre dati è quello di immagazzinarli in istruzioni DATA nel programma e poi leggere (READ) gli elementi nel programma stesso dalle istruzioni DATA. Generalmente, i dati sono immagazzinati alla fine del programma, o del segmento di programma.

Ogni volta che il computer arriva ad eseguire l'istruzione READ, prende l'elemento successivo dei dati dall'istruzione DATA e lo pone nella locazione

specificata nell'istruzione READ.

Perché sia eseguita un'istruzione READ, ci deve essere un elemento corrispondente DATA disponibile.

Perciò, in questo programma, accade quanto segue:

```
10 READ A$
20 READ B$
30 READ C$
100 DATA FRANCO,GIUSEPPE
110 DATA STEFANO
```

Programma 8

alla linea 10, READ dice al computer di prendere il primo elemento di DATA e memorizzarlo nella locazione A$. Il primo elemento di DATA si trova nella linea 100 ed è FRANCO, così FRANCO va nella locazione A$. Alla successiva istruzione READ (20), il computer prende il successivo elemento DATA che è GIUSEPPE, e così via. Puoi controllare quanto è accaduto, mettendo nel programma:

```
40 PRINT A$
50 PRINT B$
60 PRINT C$
```

RUN
FRANCO
GIUSEPPE
STEFANO

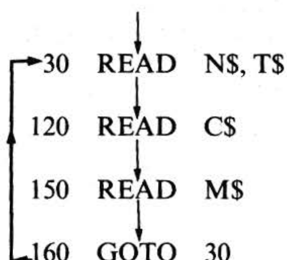
DAV7

Cosa c'è di sbagliato in questo segmento di programma?

```
10 READ A$
20 READ B
30 READ C
40 READ D$
50 DATA PAOLO,MARINA,63
```

Programma 9

Un esempio più complesso è illustrato da quest'altro segmento di programma:




```
200 DATA LUCA,ANTONIO,FRANCO
210 DATA PAOLO,GIUSEPPE
220 DATA MARIO
230 DATA STEFANO,ANDREA,MAURO,DANIELE
240 DATA MORENO,MARCO
```

Ecco cosa succede:

Primo giro del programma

N$ legge LUCA
T$ legge ANTONIO
C$ legge FRANCO
M$ legge PAOLO

Memoria alla fine del primo giro

N$	LUCA
T$	ANTONIO
C$	FRANCO
M$	PAOLO

Secondo giro del programma

N$ legge GIUSEPPE
T$ legge MARIO
C$ legge ROSSI
M$ legge VERDI

Memoria alla fine del secondo giro

N$	GIUSEPPE
T$	MARIO
C$	STEFANO
M$	ANDREA

DAV8

Quale sarà lo stato finale della memoria quando tutti i dati saranno stati letti?

DAV9

Quali sarebbero i contenuti delle locazioni A$, B$, e C$, dopo che questo segmento di programma ha letto tutti gli elementi di DATA?

```
5 REM ** DAV 9**
10 READ A$
15 REM **INTRODUZIONE DATI**
20 READ B$
30 READ C$
40 GOTO 15
45 REM **********
50 DATA AMBULANTE,CACCIATORE,SOLDATO
60 DATA MARINAIO,UOMO RICCO
```

Programma 10

RESTORE

Il BASIC ha un'altra istruzione associata all'istruzione READ: RESTORE. Quando il calcolatore incontra RESTORE, non accade come nel caso di DATA letto da READ; la successiva istruzione READ tornerà indietro e leggerà il primo elemento della lista DATA ancora una volta e poi continuerà a leggere gli elementi successivi della lista. Questo fatto può essere molto utile se vuoi usare gli stessi dati più di una volta nel corso di un programma, ma battendoli solo una volta.
Bisogna fare attenzione che il numero dei READ e il numero degli elementi di DATA da leggere siano corrispondenti, altrimenti sullo schermo apparirà un messaggio OUT OF DATA che fermerà l'esecuzione del programma.

File e record

Abbastanza spesso, ci capita di voler registrare dati particolari, per es. i dati di un archivio di informazioni. Un elenco telefonico è un buon esempio di cosa sia un archivio, o come si dice in linguaggio tecnico, un file: un insieme, cioè, di registrazioni di record simili. Ogni record ha la forma

Nome	Indirizzi	Numero di telefono

e consiste di un certo numero di campi, in questo caso tre: nome, indirizzo e numero di telefono. Un record è allora un insieme di campi ed un file è un insieme di record. Un elenco telefonico è organizzato in ordine alfabetico di cognomi e questo gli conferisce una struttura semplice.

Confronto tra stringhe

In alcuni casi, ci potrebbe capitare di dover confrontare delle stringhe. Supponiamo, ad es., di avere sul nostro microcomputer un elenco telefonico personale e di voler sapere se il cognome ROSSI si trova in un record. Il computer dovrà confrontare la stringa ROSSI con tutte le altre stringhe nel campo dei cognomi del tuo elenco. E' una cosa semplice a farsi, perché ogni lettera è rappresentata dentro il computer da un codice binario.
Così

A è 100 0001
B è 100 0010

e così via (l'appendice di questa unità didattica contiene la lista di tutti i codici binari).
Perciò, le parole messe in ordine alfabetico su carta saranno rappresentate nel computer da codici in ordine numerico.

Quindi, se:

A$ = CANE
B$ = GATTO
C$ = CANE
D$ = PESCE
E$ = CANI
A$ = C$

ma B$ > A$ (è il successivo in ordine alfabetico) e E$ > A$ (la I del plurale lo colloca dopo cane, in ordine alfabetico).

Torniamo ai nostri esempi.

Esempio 2

Definisci un file i cui dati consistono di cognomi e numeri telefonici corrispondenti. Scrivi un programma BASIC che ricerchi, nel file, un cognome particolare e che se lo trova, stampi il numero telefonico associato.

Soluzione

Potremo tentare di formulare un algoritmo descrittivo come segue:
1. Avvio
2. Immetti il nome richiesto
3. Leggi il record successivo del file di dati (es. cognome e numero)
4. Se hai raggiunto la fine del file (con il cognome = "ZZZZ"), allora scrivi il messaggio "nel file non c'è" e vai a 7, altrimenti vai a 5.
5. Se il cognome richiesto = cognome nel file, allora stampa il cognome e il numero telefonico corrispondente e vai a 7, altrimenti vai a 6.
6. Torna a 3 per il record successivo.
7. Stop

Comunque, il BASIC, generalmente, non permette istruzioni complicate come 4 e 5, perciò dobbiamo suddividerle in altre istruzioni, come ti mostriamo nel seguente algoritmo:

1. Inizio
2. Inserisci il cognome richiesto
3. Leggi il record successivo dal file di dati
4. Se hai raggiunto la fine del file vai a 7, altrimenti vai a 5
5. Se il cognome richiesto = cognome nel file, allora vai a 9 altrimenti vai 6
6. Torna a 3 per il record successivo
7. Stampa il messaggio "nel file con c'è"
8. Stop
9. Stampa cognome e numero telefonico
10. Stop

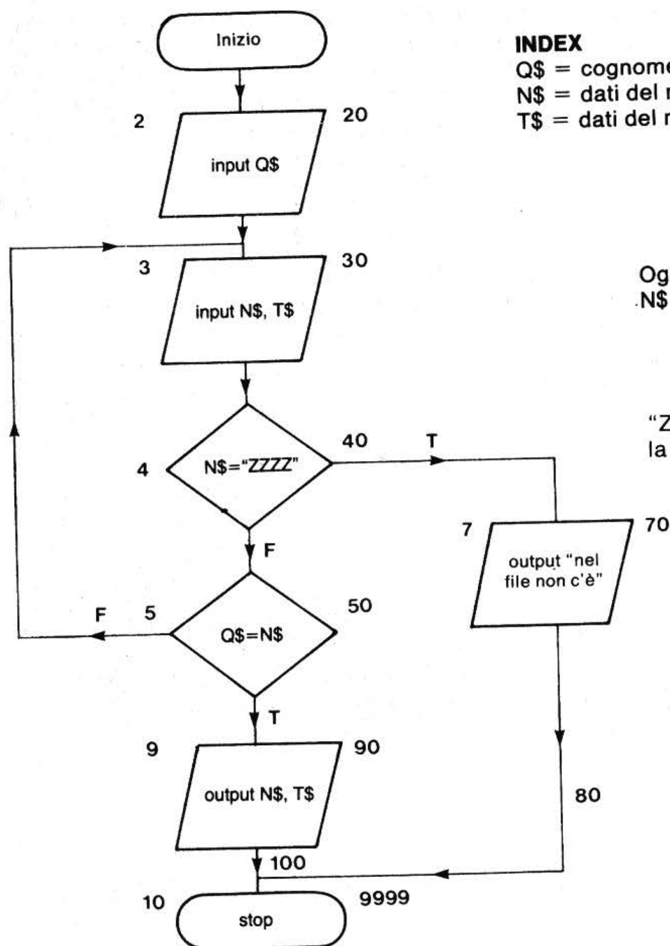


Figura 3 — Diagramma di flusso del programma.

Adesso, ogni record contiene 2 campi, in questo caso, cognome e numero telefonico.

Campo 1	Campo 2
Nome N$	Numero di telefono
Nicoletta	1234

Quindi ogni elemento di DATA deve contenere informazioni per ogni campo. Pertanto la linea READ sarà:

READ N$, T$

e le linee DATA saranno della forma:

DATA BRUNI, 1234

Nota che il numero di fine di file (END OF FILE) è un elemento di DATA (linea 310) siccome però dobbiamo utilizzare 2 locazioni (N$ e T$) è necessario aggiungere un altro elemento in DATA.
Senza questo, sullo schermo apparirebbe un messaggio di errore (out of data). Perciò scriviamo:

DATA ZZZZ, END OF FILE

e non solo

DATA ZZZZ

Questa è una stringa aperta perchè contiene degli spazi. Alcune versioni del BASIC richiedono che questa stringa venga racchiusa tra virgolette.

```
10 REM **ELENCO TELEFONICO**
15 PRINT CHR$(147)
16 PRINT"ELENCO TELEFONICO"
18 PRINT
20 INPUT"COGNOME RICHIESTO";Q$
25 REM **READ**
30 READ N$,T$                          ——— READ
40 IF N$="ZZZZ" THEN 70
50 IF Q$=N$ THEN 90
60 GOTO 30
65 REM **NOME FUORI LISTA**
70 PRINT Q$;" NON C'E' NEL FILE"
80 GOTO 9990
85 REM **NOME TROVATO**
90 PRINT"IL NUMERO DI ";Q$;" E' ";T$
100 GOTO 9990
190 REM **ELENCO DATI**
200 DATA BERETTA,1234
210 DATA CEREDA,9823
220 DATA DELISO,1850
230 DATA FAVERATO,7294
240 DATA FERRARI,5821
250 DATA GASPARELLO,4539
260 DATA PACCHETTI,7830                ——— DATA
270 DATA RISANI,1383
280 DATA ROSSI,1147
290 DATA SPINELLI,5529
300 DATA TAMBURRINI,9936
310 DATA ZZZ
9990 REM **FINE DATI**
9999 END
```

Programma 11 – Elenco telefonico

Esecuzione tipica

```
RUN
ELENCO TELEFONICO

COGNOME RICHIESTO? ROSSI
IL NUMERO DI ROSSI E' 1147

READY.
```

```
RUN
ELENCO TELEFONICO

COGNOME RICHIESTO? BIANCHI
BIANCHI NON C'E' NEL FILE

READY.


RUN
ELENCO TELEFONICO

COGNOME RICHIESTO? FERRARI
IL NUMERO DI FERRARI E' 5821

READY.


RUN

ELENCO TELEFONICO

COGNOME RICHIESTO? FERRERI
FERRERI NON C'E' NEL FILE

READY.
```

Nella quarta esecuzione abbiamo immesso il nome Ferreri, mentre nel file c'è il nome Ferrari. Il computer confronta questi insiemi di caratteri e non li trova uguali. Se cerchiamo in un elenco telefonico, potremmo renderci conto molto presto che in realtà anzichè Ferreri cerchiamo Ferrari. Naturalmente potremmo far girare di nuovo il programma con una serie di nomi simili, se avessimo qualche dubbio.

- ESEGUI IL PROGRAMMA 11 -

DAV 10

Quali cambiamenti devi fare al programma 11 per immettere il numero di telefono di una persona ed ottenere il cognome o il messaggio "non c'è nel file"?

3.8 Ordinamento

Avrai notato che, nell'esempio 2, i dati dell'elenco telefonico erano in ordine alfabetico, così come ti saresti aspettato. Sarebbe difficile per l'utente utilizzarlo se non fosse così. Comunque, la nostra soluzione al problema di ricerca non ricorreva a questo tipo di informazione: abbiamo cercato nel file di dati, record per record finché abbiamo trovato il cognome, o raggiunto la fine del file.
Il nostro algoritmo avrebbe funzionato ugualmente anche se i dati non si fossero trovati in ordine alfabetico. Più avanti, ci occuperemo di ordinare e ricercare i dati e a quel punto scoprirai i vantaggi dell'ordinamento alfabetico dei dati.
Cominciamo con un problema molto semplice.

Esempio 3

Scrivi un programma BASIC per immettere due nomi nel computer ed emettere il primo nome in ordine alfabetico.

Soluzione

Algoritmo descrittivo:
1. Inizio
2. Inserisci il primo nome
3. Inserisci il secondo nome
4. Se il primo nome < del secondo nome, allora vai a 7, altrimenti vai a 5
5. Visualizza il secondo nome
6. Vai a 8
7. Visualizza il primo nome
8. Stop

La figura 4 illustra un diagramma di flusso schematico per la soluzione di questo problema.

```
100 PRINT CHR$(147)
110 REM **ORDINE ALFABETICO**
115 PRINT"PROGRAMMA DI ORDINAMENTO ALFABETICO"
120 INPUT"PRIMO NOME";A$
130 INPUT"SECONDO NOME";B$
```

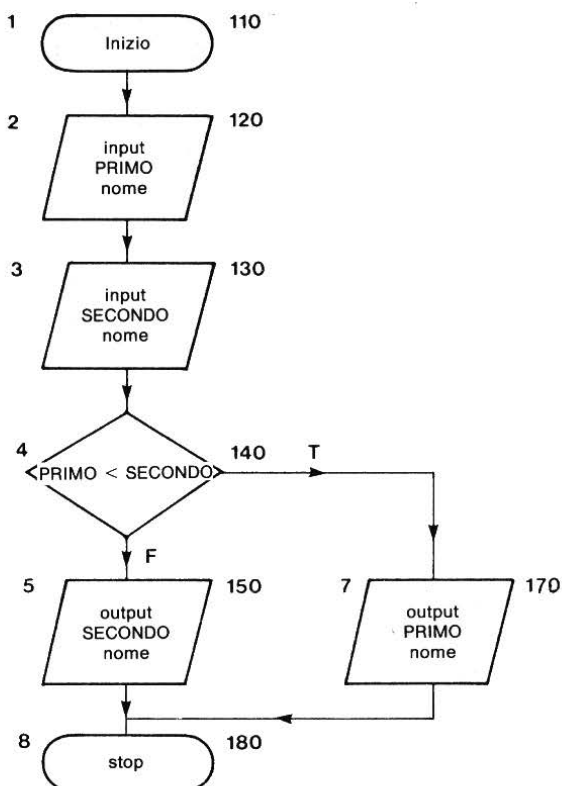


Figura 4 - Ordine alfabetico fra due nomi.

```
135 REM **CONFRONTO**
140 IF A$<B$ THEN 170
150 PRINT"IL PRIMO IN ORDINE ALFABETICO E' ";B$
160 GOTO 180
170 PRINT"IL PRIMO IN ORDINE ALFABETICO E' ";A$
180 END
```

Programma 12

Esecuzione tipica

```
RUN
PROGRAMMA DI ORDINAMENTO ALFABETICO
PRIMO NOME? BRUNO
```

```
SECONDO NOME? FERRUCCIO
IL PRIMO IN ORDINE ALFABETICO E' BRUNO

READY.

RUN
PROGRAMMA DI ORDINAMENTO ALFABETICO
PRIMO NOME? FERRUCCIO
SECONDO NOME? BRUNO
IL PRIMO IN ORDINE ALFABETICO E' BRUNO

READY.
```

- ESEGUI IL PROGRAMMA 12 -

Il gioco delle 3 carte

Supponiamo ora di voler inserire tre nomi e visualizzare quello che viene per primo in ordine alfabetico; una soluzione standard a questo problema sarebbe quella di seguire l'approccio illustrato in figura 5.

Quando chiediamo a degli studenti di risolvere questo problema, la maggior parte risponde in modo simile all'algoritmo di fig. 5. Si tratta di una soluzione buona, che però non tiene conto di problemi che potrebbero sorgere in futuro: in n particolare, infatti, abbiamo dovuto utilizzare 3 decisioni e 3 funzioni di output. Questo metodo metterebbe a dura prova la nostra pazienza e la nostra ingenuità se tentassimo di ripeterlo per 4, 5, mettiamo anche solo 10 volte. Il problema fondamentale è far sì che ogni variabile mantenga la sua locazione di memoria individuale e come meglio etichettare quella locazione. Una soluzione più semplice è quella di immettere i nomi uno per uno e immagazzinare quello con il valore più basso in A$.
Il programma contiene solo il dato con valore più basso e distrugge gli altri dati. Nel prossimo esercizio ti suggeriamo di provare a risolvere il problema con questo metodo.

Esercizio 3

Scrivi un programma BASIC per immettere 3 nomi ed ottenere il primo nome in ordine alfabetico, usando il metodo appena discusso.

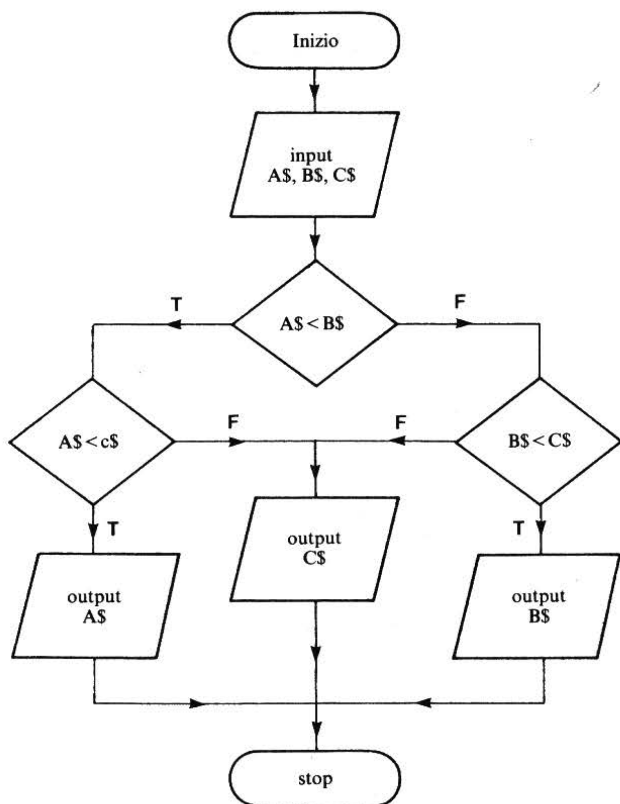


Figura 5 — Ordine alfabetico fra tre nomi.

Esercizio 4

Qui sotto ti proponiamo un archivio i cui dati sono costituiti dai nomi dei paesi europei e delle loro capitali. Scrivi un programma BASIC che utilizzi questo archivio come base di un quiz: presentando ad un utente immaginario il paese, gli si chiede qual è la capitale. Il programma deve rispondere all'input dell'utente dicendogli se ha sbagliato o no e, in questo caso, fornirgli il nome corretto.

```
140 DATA FRANCIA,PARIGI
150 DATA GERMANIA OVEST,BONN
160 DATA PAESI BASSI,AMSTERDAM
170 DATA POLONIA,VARSAVIA
180 DATA ITALIA,ROMA
190 DATA SPAGNA,MADRID
200 DATA PORTOGALLO,LISBONA
210 DATA UNGHERIA,BUDAPEST
220 DATA DANIMARCA,COPENHAGEN
230 DATA NORVEGIA,OSLO
240 DATA ZZZZ,END OF FILE
```

Compito 3

1. Progetta un algoritmo descrittivo e traccia un diagramma di flusso relativo al programma 12.

2. Modifica il programma dell'esercizio 4 in modo che conti le risposte corrette e scorrette e indichi i punteggi ottenuti alla fine del quiz.

3. Traccia un algoritmo e scrivi un programma BASIC che:
(a) immagazzini parole come lettere singole in istruzioni DATA. Per esempio, la parola "algoritmo" potrebbe essere immagazzinata come:

900 DATA A, L, G, O, R, I, T, M, O

(b) conti il numero di vocali e consonanti contenute nelle parole ed emetta i due totali e il loro rapporto (numero totale di vocali/numero totale di consonanti).

Obiettivi dell'unità 3

Ora che hai completato questa unità, controlla se sei capace di utilizzare in programmi semplici:

Locazioni di memoria per stringhe ☐
PRINT...;... ☐
Semplici procedure di salto ☐
INPUT"...";... ☐
READ ☐

DATA (un solo campo) ☐
IF A$ = B$ THEN... ☐
DATA (più di un campo) ☐
Semplici procedure di ordinamento ☐

Risposte alle DAV e agli esercizi

DAV1

Locazioni di stringhe valide: A$, T7$.
Degli altri, M8 e B9 sono locazioni di memoria per numeri.
Il resto non sono locazioni di memoria nè per numeri nè per stringhe.

DAV2

(a), (b) e (e) sono corrette ma (e) non è accettabile in un BASIC minimale.
(c) è scorretto: M$ è una locazione stringa perciò 9583 deve essere racchiuso tra virgolette:
LET M$ = "9583"
(d) è scorretto: K8 è una locazione di memoria numerica, e non gli si può assegnare la stringa "LUCA ROSSI"

DAV3

I	M	P	A	G	I	N	A	Z	I	O	N	E							
B	A	S	I	C						B	A	S	I	C					
B	A	S	I	C	B	A	S	I	C	B	A	S	I	C	B	A	S	I	C
B	A	S	I	C		B	A	S	I	C		B	A	S	I	C			

DAV4

```
10 LET B$="BASIC"
20 PRINT B$,B$
30 PRINT B$;B$;B$;B$
40 PRINT B$;"  ";B$;"  ";B$
50 END
```

Programma 13

DAV5

```
RUN
COME TI CHIAMI? ELENA VERRI
QUANTI ANNI HAI? 22
MI CHIAMO ELENA VERRI ED HO 22 ANNI

READY.
```

DAV6

```
10 INPUT"COME TI CHIAMI";N$
20 INPUT"QUANTI ANNI HAI";A
30 PRINT"MI CHIAMO ";N$;" ED HO ";
40 PRINT A;"  ANNI"
```

Programma 14

Esercizio 1

Gli esercizi 1 e 2 sono molto simili, non abbiamo pertanto riportato la risposta dell'esercizio 1

Esercizio 2

```
10 REM **CONTENUTO LETTERE**
20 PRINT CHR$(147)
30 PRINT"INFORMAZIONI SUL DESTINATARIO"
40 PRINT
50 INPUT"NOME...";N$
60 INPUT"INDIRIZZO...";S$
70 INPUT"CITTA'...";T$
80 PRINT
90 INPUT"DATA DELLA LETTERA";D$
100 PRINT
110 PRINT"DETTAGLI PRODOTTI/SERVIZI"
120 PRINT
130 PRINT"ELEMENTI DI INTERESSE"
140 INPUT I$
150 PRINT"ANNUNCIO APPARSO SU"
160 INPUT A$
170 INPUT"DATA DELL'ANNUNCIO";E$
180 PRINT
```

```
190 PRINT
200 PRINT
210 PRINT
220 PRINT N$
230 PRINT S$
240 PRINT T$
250 PRINT
260 PRINT
270 PRINT D$
275 PRINT
277 PRINT
280 FOR Z=1 TO 2000
290 NEXT Z
295 REM **LETTERA**
300 PRINT"EGREGIO SIGNORE,"
310 PRINT
315 PRINT
320 PRINT"LA PREGO DI INVIARMI INFORMAZIONI SUL"
330 PRINT I$
340 PRINT"PUBLICIZZATO SUL"
350 PRINT A$
360 PRINT"DEL GIORNO ";E$;"."
370 PRINT
380 PRINT
390 PRINT"DISTINTI SALUTI"
400 PRINT
410 PRINT
420 PRINT
430 PRINT"MARIO ROSSI"
440 END
```

Programma 15

Esecuzione tipica

```
RUN
INFORMAZIONI SUL DESTINATARIO

NOME...? GIORGIO FERRARI EDITORE
INDIRIZZO...? VIA GARIBALDI 12
CITTA'...? BOLOGNA

DATA DELLA LETTERA? 30 GIUGNO 1984
```

```
DETTAGLI PRODOTTI/SERVIZI

ELEMENTI DI INTERESSE
?CATALOGO DELLE NOVITA' EDITORIALI
ANNUNCIO APPARSO SU
? CORRIERE DELLA SERA
DATA DELL'ANNUNCIO? 28 GIUGNO 1984


GIORGIO FERRARI EDITORE
VIA GARIBALDI 12
BOLOGNA


30 GIUGNO 1984


EGREGIO SIGNORE,


LA PREGO DI INVIARMI INFORMAZIONI SUL
CATALOGO DELLE NOVITA' EDITORIALI
PUBBLICIZZATO SUL
CORRIERE DELLA SERA
DEL GIORNO 28 GIUGNO 1984.


DISTINTI SALUTI

MARIO ROSSI

READY.
```

DAV7

La linea 20 proverà a leggere il secondo elemento di DATA, MARINA, che è una stringa, ma la locazione nella linea 20 è una locazione di numero. Il computer si ferma ed indica un errore di sintassi. Per leggere "MARINA", la linea 20 dovrebbe essere 20 READ B$.
La linea 40 richiama un quarto elemento di DATA ma ci sono tre elementi soltanto nella linea 50.

DAV8

N$	MAURO
T$	DANIELE
C$	MORENO
M$	MARCO

DAV9

A$ AMBULANTE
B$ MARINAIO
C$ UOMO RICCO

Nota che "UOMO RICCO" è letto come un elemento unico, poichè non c'è virgola tra le due parole. Se esegui questo programma otterrai il messaggio come "out of data at line 20".
Perché?

DAV10

Chiamando il numero telefonico A$, sono necessari i seguenti cambiamenti alle linee 20, 50, 70 e 90:

```
20 INPUT"NUMERO TELEFONICO RICHIESTO";A$
50 IF A$=T$ THEN 90
70 PRINT"IL COGNOME CORRISPONDENTE A ";A$
75 PRINT"NON C'E'"
90 PRINT"IL COGNOME CORRISPONDENTE A ";A$
100 PRINT"E' ";N$
```

Esercizio 3

Un semplice metodo per risolvere questo problema è illustrato nel programma 16. I nomi vengono inseriti uno per uno, confrontati di volta in volta e quello con valore inferiore (cioè il primo in ordine alfabetico), è sempre immagazzinato in A$. Il programma, quindi, conserva solo un elemento d'informazione, tutti gli altri dati vengono perduti.

```
10 REM **ORDINAMENTO ALFABETICO**
15 PRINT CHR$(147)
17 PRINT"PROGRAMMA DI ORDINAMENTO ALFABETICO"
20 INPUT"PRIMO NOME";A$
25 REM **INSERIMENTO SUCCESSIVO**
30 INPUT"NOME SUCCESSIVO";B$
40 IF B$="ZZZZ" THEN 85
50 IF A$<B$ THEN 65
60 LET A$=B$
65 REM **VISUALIZZAZIONE**
70 PRINT"IL PRIMO IN ORDINE ALFABETICO E'"
75 PRINT A$
80 GOTO 25
85 REM **RISPOSTA FINALE**
90 PRINT A$;" E' IL PRIMO FRA TUTTI"
100 END
```

Programma 16

```
RUN
PROGRAMMA DI ORDINAMENTO ALFABETICO
PRIMO NOME? TOMMASO
NOME SUCCESSIVO? STEFANO
IL PRIMO IN ORDINE ALFABETICO E'
STEFANO
NOME SUCCESSIVO? GIOVANNI
IL PRIMO IN ORDINE ALFABETICO E'
GIOVANNI
NOME SUCCESSIVO? PIETRO
IL PRIMO IN ORDINE ALFABETICO E'
GIOVANNI
NOME SUCCESSIVO? FEDERICO
IL PRIMO IN ORDINE ALFABETICO E'
FEDERICO
NOME SUCCESSIVO? BARTOLOMEO
IL PRIMO IN ORDINE ALFABETICO E'
BARTOLOMEO
NOME SUCCESSIVO? ROBERTO
IL PRIMO IN ORDINE ALFABETICO E'
BARTOLOMEO
NOME SUCCESSIVO? ALBERTO
IL PRIMO IN ORDINE ALFABETICO E'
ALBERTO
```

```
NOME SUCCESSIVO? ZZZZ
ALBERTO E' IL PRIMO FRA TUTTI

READY
```

Esercizio 4

```
10 REM **CAPITALI EUROPEE**
15 PRINT CHR$(147)
20 PRINT"INDOVINA LE CAPITALI!!"
25 REM **READ**
30 READ C$,T$
40 IF C$="ZZZZ" THEN 125
50 PRINT"QUAL'E' LA CAPITALE DI ";C$
60 INPUT A$
70 IF A$=T$ THEN 105
80 PRINT"PECCATO, HAI SBAGLIATO! LA CAPITALE DI"
90 PRINT C$;" E' ";T$
100 GOTO 25
105 REM **RISPOSTA ESATTA**
110 PRINT"GIUSTO"
120 GOTO 25
125 REM **FINE QUIZ**
130 PRINT"FINE DEL QUIZ"
135 END
137 REM **DATI**
140 DATA FRANCIA,PARIGI
150 DATA GERMANIA OVEST,BONN
160 DATA PAESI BASSI,AMSTERDAM
170 DATA POLONIA,VARSAVIA
180 DATA ITALIA,ROMA
190 DATA SPAGNA,MADRID
200 DATA PORTOGALLO,LISBONA
210 DATA UNGHERIA,BUDAPEST
220 DATA DANIMARCA,COPENHAGEN
230 DATA NORVEGIA,OSLO
240 DATA ZZZZ,END OF FILE
```

Programma 17

```
RUN
INDOVINA LA CAPITALE!!

QUAL'E' LA CAPITALE DI FRANCIA
? PARIGI
GIUSTO
QUAL'E' LA CAPITALE DI GERMANIA OVEST
? BERLINO
PECCATO, HAI SBAGLIATO! LA CAPITALE DI
GERMANIA OVEST E' BONN
QUAL'E' LA CAPITALE DI PAESI BASSI
? HAGUE
PECCATO, HAI SBAGLIATO! LA CAPITALE DI
PAESI BASSI E' AMSTERDAM
QUAL'E' LA CAPITALE DI POLONIA
? VARSAVIA
GIUSTO
QUAL'E' LA CAPITALE DI ITALIA
? ROMA
GIUSTO
QUAL'E' LA CAPITALE DI SPAGNA
? MADRID
GIUSTO
QUAL'E' LA CAPITALE DI PORTOGALLO
? LISBONA
GIUSTO
QUAL'E' LA CAPITALE DI UNGHERIA
? PRAGA
PECCATO, HAI SBAGLIATO! LA CAPITALE DI
UNGHERIA E' BUDAPEST
QUAL'E' LA CAPITALE DI DANIMARCA
? COPANHEEN
PECCATO, HAI SBAGLIATO! LA CAPITALE DI
DANIMARCA E' COPENHAGEN
QUAL'E' LA CAPITALE DI NORVEGIA
? OSLO
GIUSTO
FINE DEL QUIZ

READY
```

Appendice

Codice ASCII o American Standard Code for Information Interchange (codice standard americano per interscambio di informazioni).

La parte del codice che ci riguarda è illustrata di seguito:

	32	1	49
!	33	2	50
"	34	3	51
#	35	4	52
$	36	5	53
%	37	6	54
&	38	7	55
'	39	8	56
(	40	9	57
)	41	:	58
*	42	;	59
+	43	<	60
,	44	=	61
−	45	>	62
.	46	?	63
/	47	@	64
0	48	A	65
B	66	R	82
C	67	S	83
D	68	T	84
E	69	U	85
F	70	V	86
G	71	W	87
H	72	X	88
I	73	Y	89
J	74	Z	90
K	75	[	91
L	76	£	92
M	77	]	93
N	78	↑	94
O	79	←	95
P	80		
Q	81		

I caratteri da 96 a 127 e da 161 a 191 sono caratteri grafici. Da 133 a 140 sono tasti funzione. È possibile ottenere tutti i caratteri ASCII mediante il comando PRINT CHR$(n) dove n è il numero del codice ASCII relativo al carattere voluto.

UNITA' 4
Le liste

4.1 Variabili

Abbiamo già visto come, nel corso dell'esecuzione di un programma, una locazione di memoria possa contenere molti valori differenti. Il valore in una locazione di memoria può variare durante un'esecuzione e, pertanto, spesso consideriamo i nomi di locazioni come variabili in un programma. Le 286 etichette di memoria:
A, A0, A1,..., A9, B, B0,...,Z8, Z9
sono chiamate variabili numeriche e le loro controparti
A$, A0$, A1$,..., A9$, B$, B0$,...,Z8$, Z9$
Sono chiamate variabili–stringa. Le etichette di memoria sono usate nelle espressioni delle istruzioni di un programma proprio come i matematici usano le variabili nelle equazioni.

4.2 Liste

Le liste e i supermercati sembrano essere due cose legate assieme inestricabilmente. Andiamo al supermercato con una lista di oggetti che vogliamo comprare, ne usciamo con quegli oggetti e con una lista dei prezzi, scritti su uno scontrino. La lista dei prezzi è il risultato del processo di trasferimento degli oggetti dal carrello alla cassa. Questo trasferimento avviene in modo casuale ma, se vogliamo, possiamo dare alla lista un ordine più significativo, in molti modi.
Con un po' di pazienza possiamo togliere gli oggetti dal carrello in ordine di prezzo, per es. prima il meno costoso, poi quello immediatamente più caro e così via fino al più costoso, in modo che i prezzi, sullo scontrino, siano in ordine di costo.
In modo simile, possiamo aver tirato fuori dal carrello i nostri acquisti in ordine di peso, prima il più leggero e alla fine il più pesante; così facendo imponiamo un ordine completamente differente sullo scontrino. La capacità di collegare in qualche modo la posizione nella lista con il valore dell'oggetto è la caratteristica più utile delle liste, come vedremo alla fine di questa unità.

4.3 Variabili lista

La maggior parte dei dati che abbiamo considerato fino ad ora può essere classificata in insiemi di risposte a test, insiemi di nomi e numeri telefonici associati, insiemi di paesi e di loro capitali. La maggior parte dei dati possono essere classificati in modo simile. Se stiamo raccogliendo dati per un qualche

scopo, è questo scopo che dà caratteristiche comuni all'insieme di valori. Ci sono vantaggi ovvi a dare alle locazioni di memoria per gli elementi di un insieme un nome che metta in risalto il fatto che tutti gli elementi appartengono a quell'insieme. O meglio, sarebbe utile se i nomi di locazioni di memoria identificassero la posizione di un elemento all'interno dell'insieme. Per esempio, una notazione di memoria, o di variabile, la quale sottolinei che i valori sono in qualche modo associati l'uno all'altro e indichi una posizione all'interno dell'insieme. Con un esempio ti spieghiamo come è possibile fare tutto questo.

Considera un insieme di voti sul registro di un insegnante. Essi formano una normalissima lista. Nel nostro BASIC minimale potremmo assegnare delle locazioni di memoria simbolizzate nel modo seguente:

Elemento (Voto)	**Simbolo della locazione di memoria**
Il primo membro della lista V è 42	V(1)=42
Il secondo membro della lista V è 67	V(2)=67
Il terzo membro della lista V è 90	V(3)=90
etc...	

Potremmo dire che V(1), V(2), V(3) sono delle locazioni di memoria separate. Puoi prendere una qualsiasi delle 572 locazioni di memoria farla seguire da numeri tra parentesi e costituire in tal modo delle locazioni di memoria per liste. E cioè:

Nome della lista	**Locazioni di memoria per lista**
V(I)	V(1), V(2), V(3)...
A0(I)	A0(1), A0(2), A0(3)...
C$(I)	C$(1), C$(2), C$(3)...
Q6$(I)	Q6$(1), Q6$(2), Q6$(3)...

Il numero tra parentesi (qui indicato genericamente con I) è chiamato indice della lista e può essere un qualsiasi intero positivo entro il limite posto dal tuo computer. Abbiamo chiamato la nostra lista "V", dove V sta per voto, ma se la tua versione del Basic permette l'uso di nomi di variabile più lunghi, allora ti consiglieremmo di chiamarla lista "VOTO" e le variabili della lista diventerebbero VOTO (1), VOTO(2), VOTO(3),... e genericamente VOTO(I).

Liste stringa

Come puoi vedere dalla tavola riprodotta qui sopra, le liste possono essere sia liste–stringa che liste–numeriche. Perciò, se chiami una lista V(I), si tratterà di una lista di numeri, ma V$(I) è chiaramente una lista di stringhe, (ad es. una lista di nomi da memorizzare).

Indice	Elemento	Nome di variabile
1	Giorgio	N$(1), o NOME$(1)
2	Aldo	N$(2), o NOME$(2)
3	Sergio	N$(3), o NOME$(3)
etc.	etc.	etc.

Figura 1 – Nomi di liste di stringhe.

Liste, matrici e vettori

Una tabella di dati, come quella illustrata in fig. 1, è chiamata MATRICE DI DATI.
Quando i dati sono ordinati in questo modo, per righe e per colonne, la tabella risultante viene chiamata matrice bidimensionale. In modo simile, una lista (per es. una colonna di dati) è chiamata matrice monodimensionale o vettore.
Nelle Unità successive, vedremo come il BASIC tratta le matrici bidimensionali.

DIM ovvero QUANTO E' LUNGA UNA LISTA?

Lunga quanto vuoi! Abbiamo detto che l'indice può essere un qualsiasi intero positivo entro i limiti posti dalla memoria del particolare computer. Possiamo quindi scegliere liste di lunghezza qualsiasi sempre all'interno di queste limitazioni. Su alcuni microcomputer (come sul Commodore 64) non è necessario avvisare il computer che stai usando liste di 10 o meno elementi, ma con 11 o più elementi dovrai usare l'istruzione DIM (DIMENSIONE):

numero di linea DIM A (lunghezza della lista),

che deve apparire nel programma prima che sia usata la matrice A.

In questa unità parleremo delle istruzioni DIM.
Anche se il tuo computer non ha bisogno di istruzioni DIM per piccole matrici, lasciale lo stesso: ti serviranno per comprendere il funzionamento.

Elementi e numeri di indice

Ti consigliamo di svolgere gli esercizi che seguono con carta e penna.
Potrai così comprendere meglio il significato dei termini "elemento" e "indice" e della loro relazione spesso molto vaga. Questi esercizi, inoltre, costituiscono la base per affrontare più avanti, in questa stessa unità, lo studio della procedura di interscambio–ordinamento.

Esempio 1

Qui sotto ti diamo una lista che contiene cinque elementi.
Trasferisci l'elemento di valore minore nella prima posizione, confrontandolo con ognuno degli altri valori. Segui le indicazioni che ti illustriamo nella tabella.
Scambia la posizione degli elementi, se quello che stai confrontando con il primo ha valore inferiore al primo. (è più facile a farsi che a dirsi!).

Inizio		Stadi di confronto e interscambio			
posizione o indice	elemento	1° giro C	2° giro C & i	3° giro C	4° giro C & i
1	3	3 ←	−8 ←	−8 ←	−11 ←
2	42	42 ←	42	42	42
3	−8	−8	3 ←	3	3
4	9	9	9	9 ←	9
5	−11	−11	−11	−11	−8 ←

Figura 2 – Procedura per porre in prima posizione il numero con il valore più basso.

DAV1

Applica la procedura illustrata nell'esempio 1 alla seguente lista di numeri:

6, 8, 4, 7, 3, 9, 1.

4.4 Liste in INPUT ed OUTPUT

Prima di poter manipolare gli elementi di una lista dobbiamo inserirli nel computer e, dopo l'elaborazione, generalmente otterremo un'altra lista.

Esempio 2

Scrivi un programma BASIC che ti permetta di inserire tre numeri in una lista ed ottenere come risultato gli elementi della lista in ordine inverso.

Soluzione

Chiamiamo A(I) la nostra lista. Il programma che abbiamo richiesto è:

```
10 REM **ESEMPIO**
12 REM **INVERTIRE UNA PICCOLA LISTA**
13 PRINT CHR$(147)
15 DIM A(3)
20 INPUT A(1)
30 INPUT A(2)
40 INPUT A(3)
50 PRINT
60 PRINT
70 PRINT A(3),A(2),A(1)
80 END
```

Programma 1 - Invertire l'ordine di una lista.

Esecuzione tipica

```
RUN
? 29
? 32
? -17


-17          32          29

READY
```

- ESEGUI IL PROGRAMMA 1 -

Pur avendo risposto alla domanda, non abbiamo fatto un avanzamento significativo nella tecnica di programmazione: avremmo potuto risolvere il lavoro utilizzando le tecniche apprese nel corso delle prime unità, semplicemente chiamando le variabili P, Q e R ed ottenendo R, Q e P. Per fare di meglio, dovremmo numerare gli elementi della lista, nell'ordine in cui sono stati immessi, in modo da poter usare il contatore in ordine inverso quando otteniamo l'output della lista. L'esempio successivo apporta al programma questo dettaglio.

Numerare una lista

Esempio 3

Scrivi un programma che ti permetta di:
- inserire 5 numeri di una lista,
- ottenere la stampa dei valori della lista con gli indici in forma tabellare,
- ottenere infine gli elementi della lista in ordine inverso.

Soluzione

Già nella domanda, sono elencate le 3 parti principali che guidano alla soluzione del problema. Illustriamole in forma di diagramma di flusso come nella Fig. 3.

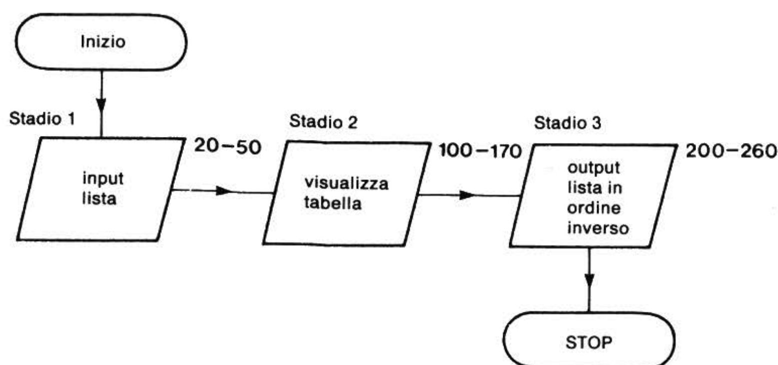


Figura 3 – Stadi per risolvere il problema posto nell'esempio 3.

Stadio 1

Usiamo la variabile C per contare gli elementi della lista in input ed in modo che faccia da indice per la lista L.

```
5 REM **LISTA IN INPUT**
10 PRINT CHR$(147)
15 DIM L(5)
20 LET C=1
25 REM **INIZIO DEL CICLO**
30 INPUT"INSERISCI IL PROSSIMO NUMERO ";L(C)
40 LET C=C+1
50 IF C<=5 THEN 25
```

Programma 2 - Contare gli elementi di una lista.

Stadio 2

La tabella sarà strutturata come quella che hai trovato nella risposta alla DAV1. Un semplice PRINT...,... sarà sufficiente a far apparire la tabella sullo schermo.

```
100 REM **VISUALIZZA TABELLA**
110 PRINT
120 PRINT
130 PRINT"INDICE","ELEMENTO"    ┐
145 REM **STAMPA TABELLA**      │
150 PRINT C,L(C)                ├──── stampa la tabella
160 LET C=C+1                   │
170 IF C<=5 THEN 145            ┘
```

Programma 3 - Stampa della lista.

Stadio 3

Ora, facciamo in modo che C conti all'indietro da 5 a 1, per stampare la lista in ordine inverso.

```
200 REM **LISTA IN ORDINE INVERSO**
210 PRINT
220 PRINT
230 LET C=5
235 REM **STAMPA**
```

```
240 PRINT L(C)
250 LET C=C-1
260 IF C<=1 THEN 240
```

stampa la lista in ordine inverso

Programma 4 Stampa in ordine inverso.

Abbiamo appena illustrato quali sono i tre moduli di programma che ci portano alla soluzione. Ora, tutto ciò che dobbiamo fare è metterli insieme come ti indicheremo. I leggeri cambiamenti che vedrai (e che non hanno effetto su ciò che fa il programma) sono spiegati nei commenti.

```
5 REM **LISTA IN INPUT**
10 PRINT CHR$(147)
12 PRINT"MANIPOLAZIONE DI UNA LISTA"
15 DIM L(5)
20 LET C=1
25 REM **INIZIO DEL CICLO**
30 INPUT"INSERISCI IL PROSSIMO NUMERO";L(C)
40 LET C=C+1
50 IF C<=5 THEN 25
60 REM *******************
70 REM *******************
100 REM **MOSTRA LA TABELLA**
110 PRINT
120 PRINT
130 PRINT"INDICE","ELEMENTO"
140 LET D=1
145 REM **STAMPA TABELLA**
150 PRINT D,L(D)
160 LET D=D+1
170 IF D<=5 THEN 145
180 REM *******************
190 REM *******************
200 REM **LISTA IN ORDINE INVERSO**
210 PRINT
220 PRINT
230 LET E=5
235 REM **STAMPA**
240 PRINT L(E)
250 LET E=E-1
260 IF E>=1 THEN 235
270 END
```

I comandi REM aiutano il lettore a capire il programma

Abbiamo usato D come indice; l'importante non è il nome ma il suo valore

Un pò di REM.

Qui abbiamo usato E

Aggiungere END.

Programma 5 - The full reverse list program.

Cambiamenti:

```
RUN
MANIPOLAZIONE DI UNA LISTA
INSERISCI IL PROSSIMO NUMERO? -8
INSERISCI IL PROSSIMO NUMERO? 15
INSERISCI IL PROSSIMO NUMERO? 23
INSERISCI IL PROSSIMO NUMERO? -4
INSERISCI IL PROSSIMO NUMERO? 19


INDICE       ELEMENTO
 1           -8
 2            15
 3            23
 4           -4
 5            19


 19
-4
 23
 15
-8

READY
```

- ESEGUI IL PROGRAMMA 5 -

Per risolvere il problema posto nell'esempio 3, abbiamo fatto un piccolo passo avanti nelle nostre conoscenze di programmazione, ma il nostro programma lavora solo con liste di cinque numeri.
Ti sembrerà un passo veramente piccolo per uno sforzo così grande!
Ma se cambiamo le istruzioni 20–50 che contano i 5 elementi, possiamo scrivere un programma che accetti un numero qualsiasi di elementi, purché prima di immetterli, sappiamo quanti siano.
Dobbiamo far sì che DIM M$ dipenda dal numero di elementi che immettiamo. Poiché il contatore C andrà fino a N+1 (guarda la linea 50), DIM M$ deve essere N+1.

```
10 REM **LISTA DI ELEMENTI**
15 PRINT CHR$(147)
```

```
20 INPUT"QUANTI ELEMENTI
   NELLA LISTA";N
25 DIM M$(N+1)
30 LET C=1
35 REM **INIZIO INSERIMENTO**
40 INPUT"INSERISCI L'ELEMENTO SUCCISSIVO";M$(C)
50 LET C=C+1
60 IF C<=N THEN 35
```

qualunque numero di elementi può essere ora inserito nella lista M$(C)

Programma 6

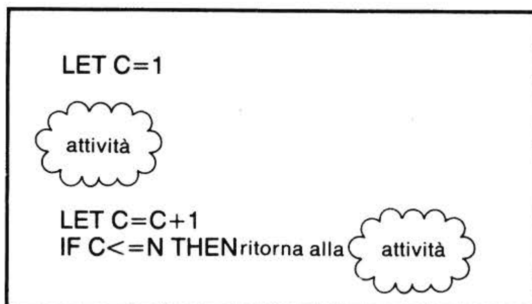
Poter immettere un qualsiasi numero di elementi in locazioni di memorie singole usando solo una dozzina di istruzioni, rappresenta un miglioramento significativo nella tecnica di programmazione. Ma noi, naturalmente, non ci sentiamo mai soddisfatti! Perché dovremmo impegnarci a contare gli elementi di una lista, specialmente se questa è lunga, quando possiamo farlo fare al computer? Affronterai questo problema nel prossimo esercizio.
(b) Modifica ora il programma per ottenere gli elementi il cui indice è dispari.

Esercizio 1

(a) Scrivi un programma BASIC per immettere una lista di numeri di lunghezza indeterminata. Poni fine alla lista con il valore terminale −9999. Chiama la lista P(C) ed assumi che tale lista contenga 30 elementi o meno. Usa perciò l'istruzione DIM P(31).
Controlla la risposta.

4.5 Il ciclo FOR...NEXT

Come abbiamo detto, il computer è capace di fare un gran numero di operazioni ripetitive. Per controllare queste operazioni, generalmente dobbiamo contarle. Da quando abbiamo introdotto il concetto di contare nell'unità 2, abbiamo usato parecchie volte la sequenza di istruzioni seguente:



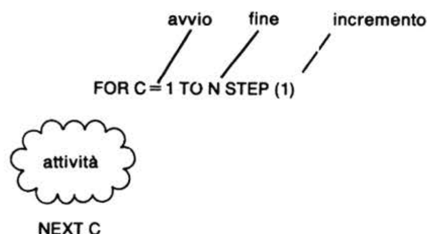
Programma 7

Queste operazioni ripetive sono tanto importanti nella programmazione che si è pensato di trovare una soluzione per facilitarne l'uso. In BASIC si usano le istruzioni FOR...NEXT...

La sequenza del programma 7 ha 3 elementi:

LET C=1	che avvia il contatore
LET C=C+1	che definisce il passo di incremento (1 in questo caso, 2 nella linea 270 della risposta all'esercizio 1)
C< =N	che ferma il processo di numerazione

Il ciclo FOR...NEXT... ha le stesse caratteristiche, per cui la sequenza precedente diventa:



Programma 8

Nota che NEXT C riporta automaticamente il controllo alla linea FOR C=1 TO N STEP (1): non hai bisogno di indicare il numero di linea di FOR C nell'istruzione NEXT C.
Matrici e cicli FOR...NEXT... sono fatti le une per gli altri.
Insieme, formano forse la più potente facilitazione del linguaggio BASIC. Vediamo, in dettaglio, come lavorano tali cicli e ripetiamo alcune delle nostre prime routine con le liste, usando queste nuove facilitazioni.

Esempi di cicli FOR...NEXT in azione

(a)

```
10 FOR I=4 TO 10 STEP(2)
20 PRINT I
30 NEXT I
RUN
 4
 6
 8
 10

READY
```

(b)

```
10 FOR K=11 TO 4 STEP(-2)
20 PRINT K
30 NEXT K
RUN
 11
 9
 7
 5

READY
```

(c)

```
10 FOR J=-3 TO 10 STEP(3)
20 PRINT J
30 NEXT J
RUN
-3
 0
 3
 6
 9

READY
```

(d)

```
10 FOR L=4 TO -5 STEP(-2)
20 PRINT L
30 NEXT L
RUN
 4
 2
 0
-2
-4

READY
```

Programmi 9-12

Il BASIC del Commodore 64 non richiede l'uso delle parentesi attorno al numero di STEP.

DAV2

Scrivi le liste di numeri stampati dai cicli seguenti:

(a)

```
10 FOR E=1 TO 9 STEP(2)
20 PRINT E
30 NEXT E
```

(b)

```
10 FOR F=-30 TO -18 STEP(3)
20 PRINT F
30 NEXT F
```

(c)

```
10 FOR G=8 TO -4 STEP(-5)
20 PRINT G
30 NEXT G
```

(d)

```
10 FOR H=-2 TO -11 STEP(-4)
20 PRINT H
30 NEXT H
```

Programmi 13-16

FOR...NEXT...con STEP (1)

L'esempio sopra riportato e le DAV, hanno usato incrementi di 2, 3, –2, –4, e –5. Abbastanza spesso, comunque, ti capiterà di dover usare semplicemente un passo 1. Quando ti trovi in questa situazione, puoi omettere STEP(1) dall'istruzione. Perciò il computer comprende che

FOR C = 1 TO N



NEXT C

Programma 17

ha un incremento di 1.

Routines di INPUT/OUTPUT con FOR...NEXT

Le routine sono più semplici da usare con le facilitazioni FOR...NEXT. Per esempio, possiamo riscrivere il programma 5 usando questi cicli. Nota che alle linee 20 e 150 richiediamo un passo di 1, perciò STEP(1) è stato omesso.

Confronto con il programma 5

```
5 REM **LISTA CON CINQUE NOMI**
10 PRINT CHR$(147)
12 PRINT"MANIPOLAZIONE DI UNA LISTA"
15 DIM L$(5)
20 FOR C=1 TO 5
25 REM **INIZIO DEL CICLO D'INSERIMENTO**
30 INPUT"QUAL'E' IL NOME SUCCESSIVO";L$(C)
40 NEXT C
50 REM ***************
60 REM ***************
```

Sostituisce le linee 20-50 (linee 20-40)

```
100 REM **MOSTRA LA TABELLA**
110 PRINT
120 PRINT
130 PRINT"INDICE","NOME"
140 PRINT
145 REM **INIZIO DEL CICLO DI STAMPA**  ┐ Sostituisce
150 FOR D=1 TO 5                        ├ le linee
160 PRINT D,L$(D)                       ┘ 140-170
170 NEXT D
180 REM ***************
190 REM ***************
200 REM **LISTA IN ORDINE INVERSO**
210 PRINT
220 PRINT
225 REM **STAMPA**
230 FOR E=5 TO 1 STEP-1
240 PRINT E,L$(E)   ┐
                    ├──── Sostituisce le linee 230-260
250 NEXT E          ┘
280 END
```

Programma 18 – Uso di FOR...NEXT per invertire una lista

```
RUN
MANIPOLAZIONE DI UNA LISTA
QUAL'E' IL NOME SUCCESSIVO? DANIELE
QUAL'E' IL NOME SUCCESSIVO? GIORGIO
QUAL'E' IL NOME SUCCESSIVO? SANDRO
QUAL'E' IL NOME SUCCESSIVO? ALBERTO
QUAL'E' IL NOME SUCCESSIVO? OSVALDO
INDICE      NOME

 1          DANIELE
 2          GIORGIO
 3          SANDRO
 4          ALBERTO
 5          OSVALDO
 5          OSVALDO
 4          ALBERTO
 3          SANDRO
 2          GIORGIO
 1          DANIELE
READY
```

- ESEGUI IL PROGRAMMA 18 -

Il ciclo FOR...NEXT

Il ciclo FOR...NEXT... può essere espresso in termini molto generali come:

```
FOR I = S TO F STEP (J)
```




```
NEXT I
```

purché ad S, F e J siano stati assegnati valori ragionevoli prima che venga eseguito il ciclo. Per chiarire meglio, valori "irragionevoli" sarebbero:

S=2, F=10, e STEP (-3)

poiché non puoi passare da 2 a 10 con passi di -3 nel normale corso degli eventi.

Esercizio 2

Nell'esercizio 2 dell'unità didattica 2 hai scritto un programma (il programma 19) che calcolava il ricavo di un investimento per un periodo di anni da specificare. Riscrivi le linee 40-90 del programma usando un ciclo FOR...NEXT...

Esercizio 3

Se hai il pallino della matematica, l'esercizio che segue e che dimostra le potenzialità del ciclo FOR...NEXT... dovrebbe piacerti.
Scrivi un programma per incolonnare i quadrati e i cubi dei numeri interi dispari da 1 a 21 incluso.

OUTPUT sullo schermo

Uno dei nostri obiettivi è quello di presentare con chiarezza i dati in output sullo schermo o sulla stampante. Le istruzioni FOR...NEXT... sono usate ampiamente nelle routine di output.

Uso di linee bianche. Come hai visto negli esempi dell'unità didattica 3,

quando scrivi una lettera è opportuno disporre le frasi in modo da lasciare qualche linea vuota. La routine seguente lo fa per te.

```
10 REM **FOR....NEXT**
15 PRINT CHR$(147)
20 REM **SALTO DI LINEE IN UNA ROUTINE DI STAMPA**
30 PRINT"CIAO"
35 REM **INIZIO DEL CICLO**
40 FOR H=1 TO 10  ┐
50 PRINT          ├── istruzione per stampare 10 linee bianche
60 NEXT H         ┘
70 PRINT"CIAO DALL'UNDICESIMA RIGA"
80 END
```

Programma 19

```
RUN
CIAO











CIAO DALL'UNDICESIMA RIGA

READY
```

- ESEGUI IL PROGRAMMA 19 -

Per tracciare una linea. Qualche volta può essere utile stampare, sullo schermo o sulla pagina, delle linee, per esempio per separare blocchi di dati o solo per sottolineare. Guarda come fa la routine seguente:

```
10 REM **FOR....NEXT**
15 PRINT CHR$(147)
20 REM **TRACCIATORE DI LINEE**
25 REM **INIZIO DEL CICLO**
30 FOR M=1 TO 40
40 PRINT"*";
50 NEXT M
60 PRINT
90 END
```

Programma 20

```
RUN
****************************************

READY
```

DAV3
Perché la linea 40 nel programma 20 ha un punto e virgola (;) alla fine della linea? Cosa succede se la linea 40 è 40 PRINT "*" (cioè senza ;)?

- ESEGUI IL PROGRAMMA 20 -

I cicli nei diagrammi di flusso

I cicli sono talmente importanti da avere un simbolo speciale sul diagramma di flusso:

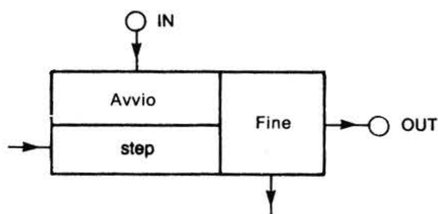


Figura 4a – Simbolo sul diagramma di flusso per il ciclo FOR...NEXT.

Le freccie senza legame sono connesse all'attività:

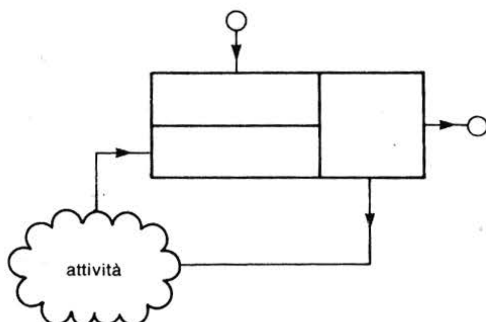


Figura 4b – Collegamenti tra i simboli del diagramma di flusso e l'attività.

4.6 Cicli "nidificati"

Il programma 20 traccia una linea di 40 asterischi. Possiamo riscrivere il programma e specificare, alla linea 30, il numero di asterischi che vogliamo e variare così la lunghezza delle linee. Con il programma:

```
10 REM **LINEE DI DIFFERENTE LUNGHEZZA**
15 PRINT CHR$(147)
20 LET N=1
30 FOR M=1 TO N
40 PRINT"*";
50 NEXT M
60 PRINT
70 END
```

Programma 21

otterremo:

```
RUN
*

READY
```

Per variare la lunghezza delle linee dovremmo semplicemente battere:
20 LET N = lunghezza desiderata

Se scriviamo:
20 LET N = 2
otterremo:

```
RUN
**

READY
```

oppure
20 LET N = 3

```
RUN
***

READY
```

oppure
20 LET N=4

```
RUN
****

READY
```

oppure
20 LET N=8

```
RUN
********

READY
```

oppure
20 LET N=16

```
RUN
****************

READY
```

oppure
20 LET N=32

```
RUN
********************************

READY
```

- ESEGUI IL PROGRAMMA 21 -

Cicli FOR...NEXT nidificati

Nel programma 21, come hai visto, si controlla l'effetto del ciclo FOR...NEXT... nelle linee 30–50, cambiando il valore di N. A questo punto di sicuro ti porrai un'ovvia domanda: perché non controllare il valore di N usando un altro ciclo FOR...NEXT...? Il programma seguente fa proprio questo. Il ciclo M delle linee 30–50 è controllato esso stesso dal ciclo N delle linee 20–70. Si dice che il ciclo M è "nidificato" nel ciclo N.

```
10 REM **CICLI NIDIFICATI FOR....NEXT**
12 PRINT CHR$(147)
15 REM **INIZIO DEL CICLO ESTERNO**
20 FOR N=1 TO 16
25 REM **INIZIO DEL CICLO INTERNO**
30 FOR M=1 TO N
40 PRINT"*";
50 NEXT M
55 REM **FINE CICLO INTERNO**
60 PRINT
70 NEXT N
75 REM **FINE CICLO ESTERNO**
80 END
```

Ciclo interno: controllo del numero di * in ogni riga

Ciclo esterno: controllo delle righe

Programma 22 - Cicli nidificati.

```
RUN
*
**
***
****
*****
******
*******
********
*********
**********
***********
************
*************
**************
***************
****************

READY
```

E' importante che tu comprenda bene come lavora questo programma. Quando, ad esempio, il computer ha finito di stampare la nona riga di asterischi N sarà uguale a 9. Il computer lascia il ciclo interno (linea 50) e il controllo passa al ciclo esterno (linea 60). Viene stampata una linea vuota e N poi incrementa il proprio valore al successivo, cioè a 10. Il controllo torna al ciclo interno, alla linea 30. Il computer ripassa poi lungo il ciclo interno (linee 30–50) per dieci volte prima di tornare alla linea 60.

- ESEGUI IL PROGRAMMA 22 -

Altri esempi di stampa con cicli

Se ti diverte generare altri esempi di stampa con cicli, qui te ne proponiamo un altro con due esercizi.

```
10 REM **CICLI NIDIFICATI FOR....NEXT**
12 PRINT CHR$(147)
14 LET P=1
15 REM **INIZIO DEL CICLO ESTERNO**
20 FOR N=1 TO 5
22 LET P=2*P
25 REM **INIZIO DEL CICLO INTERNO**
30 FOR M=1 TO P
40 PRINT"*";
50 NEXT M
55 REM **FINE DEL CICLO INTERNO**
60 PRINT
70 NEXT N
75 REM **FINE DEL CICLO ESTERNO**
80 END
```

Moltiplica il valore di P per 2 ad ogni passaggio nel ciclo

Programma 23

```
RUN
**                                      P = 2
****                                    P = 4
********                                P = 8
****************                        P = 16
********************************        P = 32

READY
```

- ESEGUI IL PROGRAMMA 23 -

Esercizio 4

Scrivi un programma usando cicli nidificati per stampare le tabelline della moltiplicazione del 7, 8 e 9.

Esercizio 5

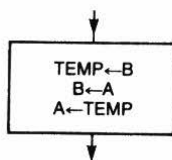
Scrivi un programma che usi cicli nidificati per stampare rettangoli di asterischi di dimensioni a tua scelta.

4.7 Interscambio

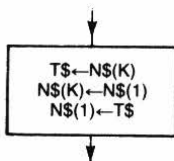
Nell'unità didattica 2 abbiamo considerato il problema di trovare il minore di due numeri e, nell'unità didattica 3, quello di trovare il minore di tre numeri. In questa unità, abbiamo fatto alcuni esercizi per scambiare gli elementi di una lista e prepararci a scrivere un programma di interscambio.
Abbiamo confrontato gli elementi di una lista con quello della posizione 1 e operato uno scambio se l'elemento nella lista era inferiore a quello nella posizione 1. Ma quando hai risposto alla DAV1, lo hai fatto manualmente. Non abbiamo ancora affrontato i problemi di un programma che esegua tale scambio. Non possiamo proprio dire

Copia A in B e poi B in A

perché la prima transazione "copia A in B" scrive sul precedente contenuto di B e lo distrugge, dandoci copie di A, in A e in B.
Invece, piuttosto che coprire B con A, dobbiamo mettere il contenuto di B in una locazione di memoria temporanea. Illustriamo questo processo nel diagramma che segue:



Dove ← significa "metti il numero della locazione a destra nella locazione a sinistra".
Supponiamo, per esempio, che vuoi ordinare un elenco di nomi della lista N$ e che vuoi scambiare il primo nome, N$(1), con un altro nome N$(K) in posizione K. E' possibile ottenere ciò usando una locazione di memoria temporanea T$:



Ricorda che sono i contenuti di N$(1) e N$(K) che vengono scambiati.
Supponiamo che N$(1) = GIANNI e N$(K) = MARIO.

Ecco cosa avviene:

	Locazione di memoria		
	N$(1)	N$(K)	T$
inizio	GIANNI	MARIO	
stadio success.	GIANNI	MARIO	MARIO
stadio success.	GIANNI	GIANNI	MARIO
fine	MARIO	GIANNI	MARIO

(Il fatto che T$ contenga ancora MARIO non è un problema: abbiamo raggiunto il nostro obiettivo, che era quello di scambiare le locazioni di GIANNI e MARIO).

Diagramma di flusso per ordinare i nomi

Nell'ordinare i numeri della DAV1, volevamo mettere il numero col valore più basso all'inizio della lista. Così, se vogliamo ordinare una lista di nomi, dobbiamo confrontare ogni nome, a turno, con quello che si trova all'inizio della lista e dobbiamo fare lo scambio solo se il nome in questione viene prima di quello all'inizio della lista.
Un diagramma di flusso potrebbe essere:

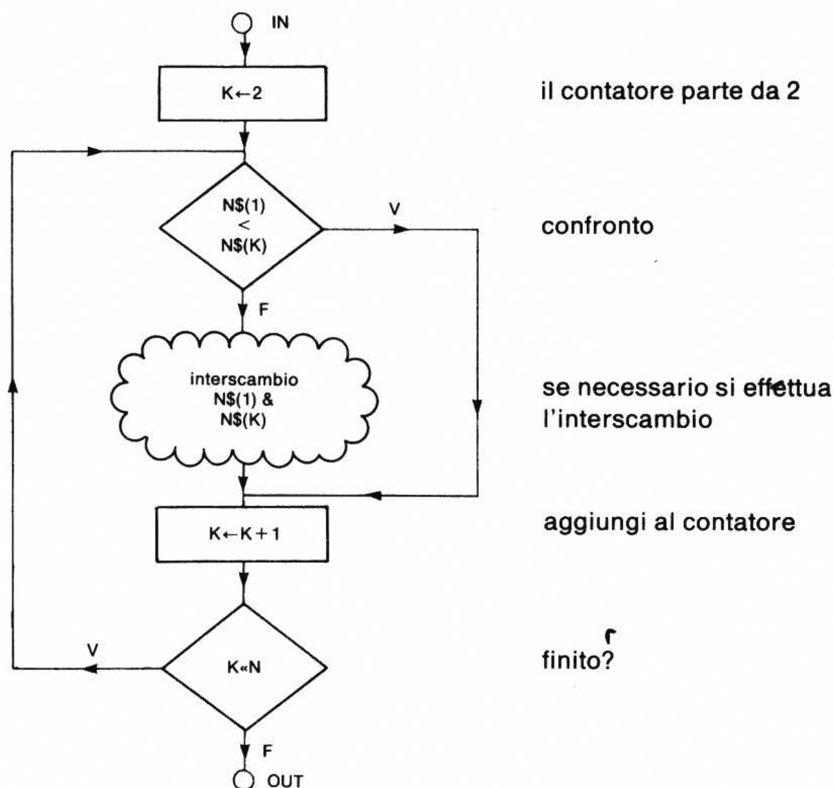


Figura 5a – Diagramma di flusso per l'interscambio.

Se vogliamo usare gli speciali simboli del ciclo FOR...NEXT... propri dei diagrammi di flusso, il diagramma appare così:

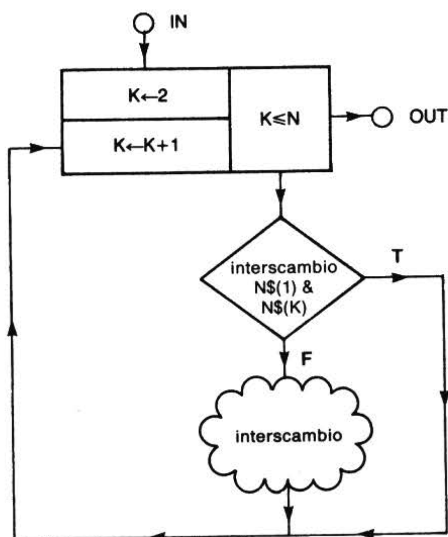


Figura 5b – Diagramma di flusso per l'interscambio con il simbolo di FOR...NEXT...

Ora questa nuova routine può essere usata per costruire un programma.

Esempio 4

Scrivi un programma

(a) che ti permetta di immettere una lista di nomi, non importa quanto lunga, in una matrice, e
(b) che stampi questa lista con l'indice in ordine di immissione;
(c) che, per mezzo della routine di interscambio metta il nome di valore alfabetico inferiore in posizione 1 nella lista e
(d) che emetta la nuova lista.

Soluzione

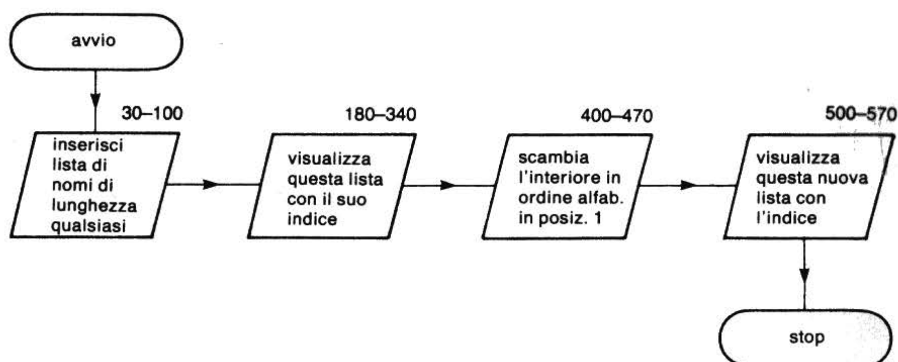


Figura 6 – Diagramma di flusso relativo all'esempio 4.

Programma di interscambio

```
10 REM **ORDINE ALFABETICO**
15 PRINT CHR$(147)
20 PRINT"PRIMO NOME IN ORDINE ALFABETICO"
25 DIM N$(30)
30 PRINT"INSERISCI UNA LISTA DI NOMI, UNO PER VOLTA"
40 PRINT"PONI FINE ALLA LISTA CON ZZZZ"
50 PRINT
60 LET I=1                                   ┐
65 REM **INSERIMENTO LISTA**                 │
70 INPUT"PROSSIMO NOME";N$(I)                ├── inserimento lista
80 IF N$(I)="ZZZZ" THEN 190                  │
90 LET I=I+1                                 │
100 GOTO 65                                  ┘
180 REM **********
190 REM **********       ┐                    trova il
200 LET N=I-1            ├─────────────────── numero di
210 REM **********       ┘                    nomi inseriti
300 PRINT
310 PRINT"INDICE","NOME"                          ┐
315 REM **INIZIO DEL CICLO DI STAMPA**            │
320 FOR J=1 TO N                                  ├ stampa lista
330 PRINT J,N$(J)                                 │
340 NEXT J                                        ┘
```

```
400 REM **********
410 REM **ROUTINE D'INTERSCAMBIO**
420 FOR K=2 TO N
430 IF N$(1)<N$(K) THEN 465
440 LET T$=N$(K)
450 LET N$(K)=N$(1)
460 LET N$(1)=T$
465 REM **SE IN ORDINE CONTINUA DA QUI**
470 NEXT
500 REM **********
505 REM **VISUALIZZA LA NUOVA LISTA**
510 PRINT
520 PRINT"LISTA DOPO L'INSERIMENTO"
530 PRINT
540 PRINT"INDICE","NOMI"
550 FOR L=1 TO N
560 PRINT L,N$(L)
570 NEXT L
580 END
```

(righe 410–470: interscambio; righe 540–570: stampa dopo l'interscambio)

Programma 24 - Ricerca del primo nome in ordine alfabetico.

Esecuzione del programma di interscambio

```
RUN
PRIMO NOME IN ORDINE ALFABETICO
INSERISCI UNA LISTA DI NOMI,
UNO PER VOLTA
PONI FINE ALLA LISTA CON ZZZZ

PROSSIMO NOME? GIORGIO
PROSSIMO NOME? PAOLO
PROSSIMO NOME? DAVIDE
PROSSIMO NOME? ERNESTO
PROSSIMO NOME? ZZZZ

INDICE     NOME
 1         GIORGIO
 2         PAOLO
 3         DAVIDE
 4         ERNESTO
```

```
LISTA DOPO L'INTERSCAMBIO

INDICE        NOME
 1            DAVIDE
 2            PAOLO
 3            GIORGIO
 4            ERNESTO

READY
```

Se, nella procedura di interscambio, viene inserita una routine come quella illustrata qui sotto (linee 470–478), possiamo vedere l'effetto di ogni passaggio attraverso il ciclo.

```
400 REM **********
410 REM **ROUTINE DI INTERSCAMBIO**
420 FOR K=2 TO N
430 IF N$(1)<N$(K) THEN 465
440 LET T$=N$(K)
450 LET N$(K)=N$(1)
460 LET N$(1)=T$
465 REM **SE IN ORDINE CONTINUA DA QUI**
472 FOR L=1 TO N
474 PRINT N$(L);" ";
476 NEXT L
478 PRINT
480 NEXT K
500 REM **********
505 REM **VISUALIZZA LA NUOVA LISTA**
510 PRINT
520 PRINT"LISTA DOPO L'INTERSCAMBIO"
530 PRINT
540 PRINT"INDICE","NOME"
550 FOR L=1 TO N
560 PRINT L,N$(L)
570 NEXT L
580 END
```

Compito 4

1. Una volta posto l'elemento di valore più basso in posizione 1, in una lista, possiamo ripetere la procedura al fine di porre tutti gli elementi della lista in ordine di valore dal più basso al più alto.
L'ordinamento della lista completa richiede l'uso di cicli FOR...NEXT...nidificati. Modifica il programma 24 per ordinare alfabeticamente una lista completa di nomi.

2. Costruisci un archivio di nomi e relativi numeri di telefono che metterai in due liste, N$(I) e T$(I) rispettivamente. Usa l'indice I per cercare nel file un nome particolare. Se il computer trova il nome, deve darti il numero di telefono associato.

Obiettivi dell'unità 4

Ora che hai completato questa unità, controlla se sei capace di scrivere semplici programmi usando:

Locazioni di memoria
- per immettere liste ☐
- per stampare liste ☐

Contatori per contare il numero di elementi in una lista. ☐

Cicli FOR...NEXT...
- per stampare una lista ☐
- per immettere una lista ☐
- per stampare serie di asterischi (*) ☐

Cicli nidificati
- per stampare serie di asterischi (*) ☐

Routine di interscambio ☐

Risposte alle DAV e agli esercizi

DAV1

I sei stadi della procedura sono illustrati nella seguente esecuzione del programma.

```
RUN
INSERISCI UNA LISTA DI NUMERI
UNO PER VOLTA
PONI FINE ALLA LISTA CON -9999

PROSSIMO NUMERO? 6
PROSSIMO NUMERO? 8
PROSSIMO NUMERO? 4
PROSSIMO NUMERO? 7
PROSSIMO NUMERO? 3
PROSSIMO NUMERO? 9
PROSSIMO NUMERO? 1
PROSSIMO NUMERO? -9999

INDICE     NUMERO
 1         6
 2         8
 3         4
 4         7
 5         3
 6         9
 7         1

 6 8 4 7 3 9 1
 4 8 6 7 3 9 1
 4 8 6 7 3 9 1
 3 8 6 7 4 9 1
 3 8 6 7 4 9 1
 1 8 6 7 4 9 3
```

Pagina mancante?

Pagina mancante?

```
17 PRINT"INTERESSI SUL CONTO BANCARIO"
20 PRINT"ANNI, DEPOSITO E % DI INTERESSE"
30 INPUT N,D,P
35 REM **********
40 FOR C=1 TO N
50 LET Y=(P*D)/100
60 PRINT"ANNO";C;"REDDITO";Y
70 LET D=D+Y
80 NEXT C
90 REM **********
100 END
```

ciclo FOR...NEXT (righe 40–80)

Programma 26

```
RUN
INTERESSI SUL CONTO BANCARIO
ANNI, DEPOSITO E % DI INTERESSE
? 5,500,11.25
ANNO 1 REDDITO 56.25
ANNO 2 REDDITO 62.578125
ANNO 3 REDDITO 69.6181641
ANNO 4 REDDITO 77.4502075
ANNO 5 REDDITO 86.1633559

READY
```

Esercizio 3

```
10 REM**QUADRATI E CUBI**
15 PRINT CHR$(147)
20 PRINT"NUMERO","QUADRATO","CUBO"
30 FOR I=1 TO 21 STEP 2
40 LET S=I*I
50 LET C=I*I*I
60 PRINT I,S,C
70 NEXT I
80 END
```

Programma 27

```
RUN
NUMERO          QUADRATO        CUBO
 1               1               1
 3               9               27
 5               25              125
 7               49              343
 9               81              729
 11              121             1331
 13              169             2197
 15              225             3375
 17              289             4913
 19              361             6859
 21              441             9261

READY
```

DAV3

Il punto e virgola (;) fa sì che, dopo la stampa di un asterisco (*), la testina di stampa si fermi e non torni a capo. Perciò, l'asterisco successivo (*) verrà stampato sulla stessa linea. Senza il punto e virgola (;) gli asterischi sarebbero stati stampati su una colonna di 40 righe.

Esercizio 4

```
10 REM **TAVOLA PITAGORICA**
20 PRINT CHR$(147)
30 PRINT"TAVOLA PITAGORICA"
40 FOR T=7 TO 9
50 FOR K=1 TO 12
60 LET P=K*T
70 PRINT K;"PER";T;"=";P
80 NEXT K
90 PRINT
100 NEXT T
110 END
```

Programma 28

```
RUN
TAVOLA PITAGORICA
 1 PER 7 = 7
 2 PER 7 = 14
 3 PER 7 = 21
 4 PER 7 = 28
 5 PER 7 = 35
 6 PER 7 = 42
 7 PER 7 = 49
 8 PER 7 = 56
 9 PER 7 = 63
 10 PER 7 = 70
 11 PER 7 = 77
 12 PER 7 = 84

 1 PER 8 = 8
 2 PER 8 = 16
 3 PER 8 = 24
 4 PER 8 = 32
 5 PER 8 = 40
 6 PER 8 = 48
 7 PER 8 = 56
 8 PER 8 = 64
 9 PER 8 = 72
 10 PER 8 = 80
 11 PER 8 = 88
 12 PER 8 = 96
```

Della tabulazione ne parleremo nella prossima unità

```
 1 PER 9 = 9
 2 PER 9 = 18
 3 PER 9 = 27
 4 PER 9 = 36
 5 PER 9 = 45
 6 PER 9 = 54
 7 PER 9 = 63
 8 PER 9 = 72
 9 PER 9 = 81
 10 PER 9 = 90
 11 PER 9 = 99
 12 PER 9 = 108

READY
```

Esercizio 5

```
5 REM **RETTANGOLO**
7 PRINT CHR$(147)
10 INPUT"LUNGHEZZA DEL RETTANGOLO";L
20 INPUT"LARGHEZZA DEL RETTANGOLO";W
30 FOR I=1 TO W
40 FOR J=1 TO L
50 PRINT"*";
60 NEXT J
70 PRINT
80 NEXT I
90 END
```

Programma 29

UNITÀ 5
Conclusioni sulle stringhe l'istruzione PRINT

5.1 Introduzione

Le precedenti unità didattiche avevano carattere introduttivo; i nuovi concetti sono stati affrontati in modo molto veloce.
In questa unità parleremo principalmente di stringhe, ma incontrerai anche l'istruzione TAB che è un'importante aggiunta al tuo repertorio di stampa. Il titolo di questa unità esagera forse un attimino, ma, alla fine, avrai incontrato la maggior parte delle stringhe principali e delle funzioni di stampa del linguaggio BASIC.

5.2 Lunghezza di una stringa di caratteri

Nell'unità didattica 3 ci siamo chiesti quanto fosse lunga una stringa. Allora, questa poteva sembrare una domanda piuttosto oziosa.
In realtà, il numero di caratteri contenuti in una particolare locazione di memoria di stringhe, costituisce spesso un'informazione di importanza vitale. In particolar modo, se stiamo cercando di usare l'allocazione di memoria di un particolare computer il più efficacemente possibile.
Nel linguaggio BASIC, l'operazione LEN(A$) ci indica la lunghezza di A$ (ovvero il numero di caratteri contenuti in A$).

Perciò:

Se A$ = "FRANCO" allora LEN(A$) = 6
se B$ = "I" LEN(B$) = 1

DAV1

Quali sono i valori di:

(a) LEN(C$) dove C$ = "ANNA"
(b) LEN(D$) dove D$ = "A"
(c) LEN (E$) dove E$ = "72"
(d) LEN (F$) dove F$ = "GATTO 123"

Esempio 1

Scrivi un programma BASIC per inserire una lista di parole, che termini con ZZZZ, e stampare la lunghezza di ogni parola.

Soluzione

Abbiamo scritto il programma per leggere la lunghezza di parole, ponendo le parole in un'istruzione DATA, in modo da ridurre il tempo di inserimento necessario. Ogni parola letta da DATA, viene messa in P$ (linea 100) e la sua lunghezza immagazzinata in L (linea 120). Poi il risultato viene stampato alla linea 140.

```
10 REM **LUNGHEZZA DI UNA PAROLA**
20 REM ********************
30 REM **LEGGE LE PAROLE DA UNA LISTA DI DATI
40 REM E NE VISUALIZZA LA LUNGHEZZA**
50 PRINT CHR$(147)
90 REM **INIZIO DEL CICLO DI LETTURA**
100 READ W$                                ┐
110 IF W$="ZZZZ" THEN 190                  │
120 LET L=LEN(W$)                          ├─ ciclo READ, LEN,
130 PRINT"'";W$;"'HA";L;"LETTERE"          │  PRINT
140 GOTO 90                                ┘
190 REM **FINE LAVORO**
200 END
910 DATA ATTENTO,AL,PEDONE,AVVELENATO,DELLA,NAJDORF
920 DATA ZZZZ
```

Programma 1 – Indica la lunghezza di parole.

```
RUN
'ATTENTO' HA 7 LETTERE
'AL' HA 2 LETTERE
'PEDONE' HA 6 LETTERE
'AVVELENATO' HA 10 LETTERE
'DELLA' HA 5 LETTERE
'NAJDORF' HA 7 LETTERE

READY
```

- ESEGUI IL PROGRAMMA 1 -

5.3 Tabelle di frequenza

Misurando la frequenza con cui un dato fenomeno accade, in generale è necessario poter manipolare un'informazione numerica. Per esempio, conoscere la frequenza con cui certe lettere occorrono nell'uso normale del linguaggio è un fattore importante nelle attività di decifrazione di codici. Per

misurare le frequenze, si può ricorrere ad una tecnica molto semplice usata nell'analisi statistica: quella di contrassegnare con una barretta, cioè di "sbarrare" gli elementi che ci interessano. Questo concetto viene introdotto, all'inizio, con un esempio a carta e penna.

Esempio 2

Ricorderai di aver sentito, da bambino, una vecchia filastrocca, che qui ti riproponiamo. Trova la frequenza di ogni vocale all'interno delle parole che si trovano nelle seguenti istruzioni DATA.

```
900 DATA LA,PIGRIZIA,ANDO',AL,MERCATO,ED,UN,CAVOLO,COMPRO'
910 DATA MEZZOGIORNO,ERA,SUONATO,QUANDO,A,CASA,RITORNO'
920 DATA PRESE,L',ACQUA,ACCESE,IL,FUOCO,SI,SEDETTE,E,RIPOSO'
930 DATA ED,INTANTO,A,POCO,A,POCO
940 DATA ANCHE,IL,SOLE,TRAMONTO',ZZZZ
```

Soluzione

Ci sono due modi diversi di affrontare il problema.

(a) Sbarrare e contare prima tutte le A, poi tutte le E, etc. In questo modo sono necessari 5 passaggi sugli stessi dati per ottenere un'informazione piuttosto scarsa

(b) disegnare una tabella e considerare ogni vocale in sequenza:

LA: sbarra la A e metti un segno sulla riga A;
PIGRIZIA: metti un segno sulla riga I, poi altri due segni sulla riga I, seguiti da un segno sulla riga A
ANDO': metti un segno in A e uno in 0.

Vocale	Conteggio	Totale
A	●●●●● ●●●●● ●●●●● ●●●●●	20
E	●●●●● ●●●●● ●●●●●	15
I	●●●●● ●●●●●	10
O	●●●●● ●●●●● ●●●●● ●●●●● ●●●●● ●	26
U	●●●●●	5

```
900 DATA LA,PIGRIZIA,ANDO',AL,MERCATO,ED,UN,CAVOLO,COMPRO'
910 DATA MEZZOGIORNO,ERA,SUONATO,QUANDO,A,CASA,RITORNO'
920 DATA PRESE,L',ACQUA,ACCESE,IL,FUOCO,SI,SEDETTE,E,RIPOSO'
930 DATA ED,INTANTO,A,POCO,A,POCO
940 DATA ANCHE,IL,SOLE,TRAMONTO',ZZZZ
```

Figura 1 – Conto completo delle vocali sbarrate.

DAV2

Usa il metodo sopra indicato per scrivere una tabella di frequenza delle lunghezze delle parole sull'esempio 2.

Come far contare il computer

Abbiamo trovato un metodo per contare le frequenze con carta e penna; ora abbiamo bisogno di un metodo per far sì che il computer faccia questi conti all'interno di una lista. Il potere delle liste deriva da un uso appropriato dell'indice. Nella domanda 2 del compito 4 abbiamo visto come gli elementi di due liste di dati (nomi e numeri di telefono) erano collegate da un indice comune. Il terzo elemento della lista–numeri corrispondeva al numero di telefono del terzo nome della lista-nomi, eccetera. Generalmente, il membro n-esimo della lista-nomi è collegato al membro n-esimo della lista-numeri. Supponiamo di voler contare il numero di volte in cui troviamo le cifre 0, 1, 2, ...9 in una sequenza. Possiamo usare 10 contatori:

C(0), C(1), C(2) ... C(9)

ognuno dei quali, all'inizio, avrà valore zero. Per contare le cifre in 473808, prendiamo la prima cifra nella sequenza: 4. Mettiamo 1 in C(4) e così via:

Cifre inserite	Contatore dopo l'inserimento									
	C(0)	C(1)	C(2)	C(3)	C(4)	C(5)	C(6)	C(7)	C(8)	C(9)
inizio	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0
7	0	0	0	0	1	0	0	1	0	0
3	0	0	0	1	1	0	0	1	0	0
8	0	0	0	1	1	0	0	1	1	0
0	1	0	0	1	1	0	0	1	1	0
8	1	0	0	1	1	0	0	1	2	0

Il concetto, quindi, è che, quando l'elemento I della lista viene immesso, C(I) viene incrementato di 1. Tutto questo può essere fatto con due sole istruzioni BASIC.

```
120 INPUT I                ]
140 LET C(I)=C(I)+1        ]—————— Un contatore di lista
```

DAV3

La sequenza

```
120 INPUT N
140 LET C(N)=C(N)+1
```

è usata per contare il numero di zeri (0), di uno (1), di due (2), ecc. nei seguenti dati in input:

3, 1, 0, 5, 9, 9, 6, 6, 6, 0, 4, 4, 2, 4, 1, 2, 1, 3, 0, 2, 1, 3.

Quali sono i valori di:

(a) C(3) dopo l'immissione di 3 numeri;
(b) C(9) dopo l'immissione di 12 numeri;
(c) C(1) dopo l'immissione di tutti i numeri;
(d) C(0) dopo l'immissione di tutti i numeri;

Nel prossimo esempio, useremo questo metodo di conteggio all'interno di una lista.

Esempio 3

Scrivi un programma per immettere una sequenza di singole cifre ed ottenere la frequenza di ogni cifra.

Soluzione

La cifra può essere una qualsiasi dell'insieme di 10 numeri 0, 1, 2, 3 ... 9. Immetteremo questi numeri uno per uno e faremo terminare la sequenza con –9999. Si tratta proprio di un lavoro di routine!

La lista di conteggio avrà 10 contatori:

C(0), C(1), C(2) ... C(9).

Il programma per risolvere il problema è costituito da 2 parti.
(i) la routine di input e di incremento contiene le due istruzioni 120 e 140 appena viste;
(ii) per la routine di output abbiamo utilizzato il ciclo FOR...NEXT, con l'indice J che va da 0 a 9.

```
10 REM **FREQUENZA DELLE CIFRE INSERITE**
20 REM **E IMMAGAZZINATE IN UN CONTATORE DI LISTA**
30 REM ***************
35 PRINT CHR$(147)
40 DIM C(10)
90 PRINT"CONTATORE DI LISTA"
100 PRINT"INSERISCI UNA ALLA VOLTA LE CIFRE DESIDERATE,"
110 PRINT"PER FINIRE DIGITA '-9999'"
120 INPUT"PROSSIMA CIFRA";I
130 IF I=-9999 THEN 190          routine di inserimento
140 LET C(I)=C(I)+1              e conto
150 GOTO 120
190 REM **TITOLO DELLA STAMPA**
200 PRINT
210 PRINT"CIFRA","FREQUENZA"
220 PRINT
230 FOR J=0 TO 9
240 PRINT J,C(J)                 stampa della tabella
250 NEXT J
260 END
```

Programma 2 – Conteggio con un contatore di lista C(I).

Output finale

(Dopo l'immissione di 3, 7, 6, 4, 9, 1, 4, 9, 2, 7, 8, 0, 1, 5, 2, 7, -9999).

```
CIFRA        FREQUENZA

 0            1
 1            2
 2            2
 3            1
 4            2
 5            1
 6            1
 7            3
 8            1
 9            2

READY
```

- ESEGUI IL PROGRAMMA 2 -

Tabella di frequenza per lunghezze di stringhe

In questa unità, abbiamo scritto due programmi: il primo per trovare la lunghezza di stringhe e il secondo per costruire una tabella di frequenze. Nell'esercizio successivo, vorremmo che mettessi assieme questi due concetti e che costruissi una tabella di frequenza delle lunghezze di parole. Se vuoi, puoi usare le parole della filastrocca che si trovano nelle istruzioni DATA, dell'esempio 2. Stabilisci che le parole non devono essere più lunghe di 15 caratteri, per cui la lista delle lunghezze conterrà gli elementi:

L(1), L(2), L(3) ... L(15).

Esercizio 1

Scrivi un programma che legga all'interno di un insieme di parole e stampi sullo schermo una tabella di frequenza delle loro lunghezze.

5.4 Diagrammi di frequenza

Diagrammi di frequenza delle vocali
La figura 1 riproduce in una tabella la frequenza delle vocali con una serie di barrette ed ha su di noi un impatto immediato: in qualche modo, ci dà – visivamente – un'informazione maggiore sulla distribuzione delle frequenze delle vocali rispetto alla colonna di cifre. Perciò, perché non far sì che il computer ci stampi un disegno? Nell'unità 4 hai già visto come si stampano linee di asterischi, guidando la testina di stampa sulla pagina (o schermo) con un ciclo FOR...NEXT... di ampiezza variabile.

DAV4

Cosa apparirà sullo schermo, come risultato del seguente programma?

```
10 READ A
20 FOR I=1 TO A
30 PRINT"*";
40 NEXT I
45 PRINT
50 GOTO 10
100 DATA 2,5,7,8,3,1
```

Programma 3

Possiamo fare la stessa cosa usando le frequenze della figura 1 per determinare il numero degli asterischi stampati sulla pagina. Verrà così generata una rappresentazione della distribuzione delle frequenze.

Esempio 4

Scrivi un programma che stampi un diagramma di frequenza per la distribuzione di vocali dell'esempio 2.

Soluzione

Nota: questo programma traccia solamente il diagramma delle frequenze già calcolate. Abbiamo immagazzinato tali frequenze nell'istruzione DATA di linea 900.
Leggiamo le frequenze (linee 50 a 80) con un contatore F(K) dove F(1) è il numero di A, F(2) è il numero di E, etc.

Stampiamo poi asterischi nella pagina, in rapporto al valore di F(K) (linee da 220 a 250).

```
10 REM **DISTRIBUZIONE DI FREQUENZA**
20 REM **LA LISTA DELLE FREQUENZE E'(K)**
30 PRINT CHR$(147)
35 DIM F(6)
40 LET K=1
45 REM **INIZIO DEL CICLO DI LETTURA**
50 READ F(K)
60 IF F(K)=-9999 THEN 110       legge le frequenze e
70 LET K=K+1                    le immagazzina in F(1), F(2),...
80 GOTO 50
90 REM *********
100 REM **NON CONSIDERA IL NUMERO -9999**
110 LET N=K-1
120 REM *********
200 REM **ROUTINE DI STAMPA**
210 PRINT
220 FOR X=1 TO N
230 FOR Y=1 TO F(X)
240 PRINT"*";
250 NEXT Y                      stampa gli asterischi
260 PRINT
270 PRINT
280 NEXT X
300 REM *********
900 DATA 9,16,10,10,6,-9999
```

Programma 4 – Disegno di una distribuzione di frequenza.

```
RUN

*********

****************

**********

**********

******


READY
```

- ESEGUI IL PROGRAMMA 4 -

Diagramma di frequenza delle lunghezze di parole

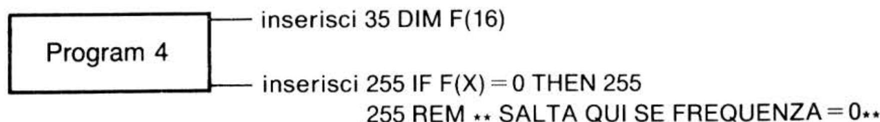
Se vogliamo disegnare un diagramma di frequenza delle lunghezze delle parole nella DAV2, abbiamo bisogno di modificare il programma 4. Sono necessari due tipi di modifiche:

– Primo: la lista di frequenza contiene più elementi. Ci sono 15 frequenze (da 1 a 15) più –9999, abbiamo quindi 16 elementi e dobbiamo aggiungere la linea 35 DIM F(16) al programma 4.

– Secondo: la routine di stampa alla linea 220 girerà anche quando la frequenza è zero. Non possiamo usare il ciclo FOR...NEXT... da 1 a 0. Perciò dobbiamo prevenire il fatto che il programma vada avanti nel ciclo FOR...NEXT quando la frequenza è zero. Aggiungiamo allora le linee

```
225 IF F(X)=0 THEN 255
255 REM*SALTA QUI SE FREQUENZA=CENTRALI*0
```

Così il programma diventa:



E naturalmente la linea 900 diventa:

```
900 DATA 1,5,4,6,9,0,1,3,1,0,0,1,0,0,0
```

Attenzione! Ricordati, il calcolatore considera sia l'accento che l'apostrofo come caratteri separati!!!

Cancella la linea 270 per adattarla a tutto lo schermo.

L'esecuzione del programma 4 è:

```
RUN

*
*****
****
******
*********

*
***
*


*



------------>-------- la stampa finisce all'incirca qui!
READY
```

Figura 2 - Diagramma di frequenza del programma 4 modificato.

- ESEGUI IL PROGRAMMA 4 MODIFICATO -

5.5. Tabulazione

Abbiamo gli elementi essenziali per tracciare un diagramma, ma siamo ancora lontani dal renderlo significativo. Ci sarebbe di aiuto la possibilità di spostare la testina di stampa ad una determinata posizione della pagina o dello schermo. Questo è ciò che, in dattilografia, viene chiamato tabulazione (disporre in forma tabulare). Nel BASIC, abbiamo la funzione TAB.
Utilizziamo lo stesso approccio dell'unità 3, per scrivere un pezzetto di programma che spieghi se stesso (un approccio che bisognerebbe adottare sempre!).

Per prima cosa, guardiamo cosa succede se numeriamo le posizioni di stampa sullo schermo:

```
50  PRINT"123456789012345678901234567890123456789 0"
60  PRINT"A";TAB(5);"E";TAB(7);"I";TAB(19);"O";TAB(31);"U"

RUN
1234567890123456789012345678901234567890
A    E I           O           U
READY.
```

Programma 5

TAB(5) ha stampato E sulla sesta posizione. Perché? Perché la macchina conta le posizioni di stampa dalla posizione 0. Te lo dimostra il programma 6 in cui la numerazione delle posizioni, sullo schermo, parte da 0:

```
50  PRINT"0123456789012345678901234567890123456789"
60  PRINT"A";TAB(5);"E";TAB(7);"I";TAB(19);"O";TAB(31);"U"

RUN
0123456789012345678901234567890123456789
A    E I           O           U
READY.
```

Programma 6

Ora TAB(5) va alla posizione indicata con 5, ma si tratta ancora della sesta posizione sullo schermo.

DAV5

Scrivi un programma per stampare COL 1, COL 2, COL 3 sullo schermo, in modo che COL 1 parta dalla posizione 0, COL 2 dalla posizione 10 e COL 3 dalla posizione 20.

Variabile TAB e suoi effetti

Con un ciclo FOR...NEXT... possiamo fare in modo che la linea 60 stampi le vocali disposte in una tabella i cui valori di tabulazione siano quelli del programma 6.

```
50 FOR I=1 TO 7
60 PRINT"A";TAB(5);"E";TAB(7);"I";TAB(19);"O";TAB(31);"U"
70 NEXT I
```

Programma 7

```
RUN
A     E I              O            U
A     E I              O            U
A     E I              O            U
A     E I              O            U
A     E I              O            U
A     E I              O            U
A     E I              O            U

READY
```

Con un altro esempio mostriamo come funziona TAB con una variabile. Se vogliamo usare TAB(V) dove V è una variabile, possiamo guidare la testina di stampa su differenti posizioni della stampante o dello schermo. Il programma:

```
30 FOR A=1 TO 10
40 PRINT TAB(A);"CIAO"
50 NEXT A
60 END
```

Il valore di A in TAB(A) è determinato dal ciclo della variabile A

Programma 8

produce:

```
RUN
 CIAO
  CIAO
   CIAO
    CIAO
     CIAO
      CIAO
       CIAO
        CIAO
         CIAO
          CIAO

READY
```

Possiamo fare un passo avanti e combinare questi due effetti in un unico programma:

```
50 FOR I=1 TO 7
60 PRINT TAB(0+I);"A";TAB(5+I);"E";TAB(7+I);"I";
65 PRINT TAB(19+I);"O";TAB(31+I);"U"
70 NEXT I
80 END
```

Programma 9

```
RUN
 A    E I          O           U
  A    E I          O           U
   A    E I          O           U
    A    E I          O           U
     A    E I          O           U
      A    E I          O           U
       A    E I          O

READY
```

DAV6

Scrivi un segmento di programma per inserire 3 numeri a tua scelta, che pongano la stringa "TITOLO" in tre diverse posizioni sulla stessa linea di output.

TAB e diagramma di frequenza

Siamo ora in grado di trasformare il diagramma di frequenza della fig. 2 in una rappresentazione più attraente.
La routine di stampa del programma 4 modificato (linee da 200 a 300) era:

```
200 REM **ROUTINE DI STAMPA**
210 PRINT
220 FOR X=1 TO N
225 IF F(X)=0 THEN 255
230 FOR Y=1 TO F(X)
240 PRINT"*"
250 NEXT Y
260 PRINT
270 PRINT
280 NEXT X
300 REM ***
```

Programma 4 (modificato)

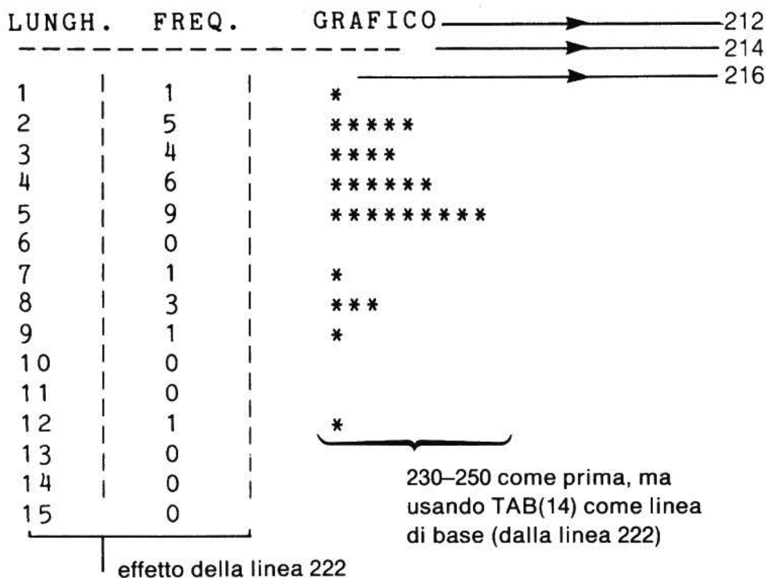
Abbiamo aggiunto:
linea 212 per stampare le intestazioni delle colonne
linea 214 per stampare una riga sullo schermo
linea 216 per stampare le divisioni tra le colonne
(vedi il ciclo seguente)
linea 222 (nel ciclo) stampa X e F (X) sulla pagina, oltre alle divisioni tra le colonne. Questa linea termina con ";" in modo che la successiva istruzione PRINT (linea 240) appaia sulla stessa linea.

Probabilmente dovrai studiare questa parte molto attentamente per controllare tutti i dettagli:

Per le linee da 10 a 120, vedi il programma 4.

```
200 REM **ROUTINE DI STAMPA**
210 PRINT
212 PRINT"LUNGH.";TAB(8);"FREQ.";TAB(14);"GRAFICO"
214 PRINT"----------------------"
216 PRINT TAB(7);"β";TAB(12);"β"
220 FOR X=1 TO N
222 PRINT TAB(2);X;TAB(7);"β";TAB(9);F(X);TAB(12);
    "β",TAB(14)
225 IF F(X)=0 THEN 255
230 FOR Y=1 TO F(X)
240 PRINT"*"
250 NEXT Y
255 REM **SALTA QUI SE FREQUENZA=0**
260 PRINT
280 NEXT X
290 REM **********************
900 DATA 1,5,4,6,9,0,1,3,0,0,1,0,0,0,-9999
```

Programma 10



- ESEGUI IL PROGRAMMA 10 -

Esercizio 2

Modifica il programma 4 per ottenere una stampa simile a quella del programma 10. Aggiungi:

(a) un cambiamento appropriato di intestazione delle colonne
(b) una stampa delle lettere A, E, I, O e U, nella colonna a sinistra
(c) alla fine del programma, la seguente riga:
...SCALA...0...5...0...5...0

5.6 Taglio di stringhe

Osserviamo ora una stringa che, sebbene sia essa stessa un'entità, contiene più di un elemento di informazione. Per esempio, 23 Giugno 1971 è un'unica data ma ci sono occasioni in cui vogliamo considerarne solo una parte, per esempio il mese.

Registrazioni di dati

Quante volte ti sei trovato a dover pagare la tassa di circolazione della tua auto e a dover riempire gli spazi riservati alla data di scadenza? o a riempire altri moduli che ti chiedevano di indicare la data in modo simile a quello che segue:

DATA						
	G	G	M	M	A	A

e cioé, scrivendo all'interno delle prime due caselle il giorno, poi il mese e, nelle due caselle finali, le ultime due cifre dell'anno?

La presentazione G G M M A A comporta dei problemi. Confronta le date 23 Giugno 1971 (230671) e 14 settembre 1973, o 140973.

L'ultima data contiene un numero più piccolo. Mentre in

4 Luglio 1933, o 040733
e
15 gennaio 1967, o 150167

l'ultima data contiene il numero maggiore. Chiaramente quindi la sequenza G G M M A A non è utile per archiviare date.

La soluzione è quella di mettere le date nella forma A A M M G G .

Quindi le 4 date precedenti diventano:
330704, 670115, 710623, e 730914

ottenendo così una maggiore chiarezza d'interpretazione.

Le date, generalmente, non sono immagazzinate nella macchina come numeri per calcoli, ma sono immesse come stringhe. La loro disposizione all'interno delle stringhe (A A M M G G) permette, nella vita di tutti i giorni, di controllare procedure che devono aver luogo e che, a loro volta permetteranno di riaggiornare la data. Facciamo un esempio:
se siamo interessati ad un aumento di salario allora le parti del numero che corrispondono all'anno e al mese potrebbero essere importanti. Se gestiamo un centro musicale e spediamo ogni tre mesi ai nostri clienti un avviso in

modo che ricordino di accordare i loro pianoforti, allora ci potrà servire soltanto il mese. Talvolta, benché l'informazione dell'intera stringa sia importante, potremmo avere un qualche buon motivo per volerne leggere solo una parte.

LEFT$(X$,I) e RIGHT$(X$,I)

Se vogliamo considerare solo parte di una stringa, allora avremo bisogno di un'istruzione che tagli la stringa stessa. Cominciamo con due tipi di istruzione.
LEFT$(X,$,I) lascia gli "I" caratteri a sinistra della stringa X$, tagliando via quelli a destra di I.

es. se X$ = "TAGLIARE"
allora LEFT$(X$,3) = TAG

l'istruzione ha cioè lasciato gli I=3 caratteri della stringa togliendo tutti quelli che si trovavano oltre il terzo carattere (LIARE)

RIGHT$(X$,I) lascia gli "I" caratteri a destra di X$.

es. RIGHT$ (X$,5) = LIARE

Vediamo ora cosa ne pensa la macchina. Immettiamo una stringa di 6 caratteri e usiamo l'indice I del ciclo FOR...NEXT... per staccare sottostringhe di lunghezza da 1 a 6. La possibilità di disporli in modo scalare (linea 30) ti aiuterà a capire cosa sta succedendo.

```
10 REM **STRINGHE**
15 PRINT CHR$(147)
20 INPUT"INSERISCI UNA STRINGA DI 6 CARATTERI";X$
30 PRINT TAB(10);"1234567890"
40 FOR I=1 TO 6
50 LET A$=LEFT$(X$,I)
60 PRINT ;I;TAB(10);A$
70 NEXT I
80 END
```

Programma 11

Esecuzione di taglio della stringa a sinistra

```
RUN
INSERISCI UNA STRINGA DI 6 CARATTERI? AB
CDEF
             1234567890
 1           A
 2           AB
 3           ABC
 4           ABCD
 5           ABCDE
 6           ABCDEF

READY
```

Ora, se cambiamo la linea 50 del programma 11 in

```
50 LET A$=RIGHT$(X$,I)
```

ed immettiamo la stringa ABCDEF, il risultato è:
Esecuzione del taglio della stringa a destra

```
RUN
INSERISCI UNA STRINGA DI 6 CARATTERI? AB
CDEF
             1234567890
 1           F
 2           EF
 3           DEF
 4           CDEF
 5           BCDEF
 6           ABCDEF

READY
```

- ESEGUI IL PROGRAMMA 11 - POI CAMBIA LA LINEA 50 USANDO RIGHT$.

DAV7

Se A$ = 1A2B3C4D, cosa sono:

(a) LEFT$(A$,1) (c) RIGHT$(A$,3)
(b) LEFT$(A$,4) (d) RIGHT$(A$,4)

Taglio di stringhe di lunghezza variabile

Il programma 11 è un po' bruttino perché abbiamo dovuto specificare (alla linea 40) quanto era lunga la stringa: 6 caratteri. Ma potremmo facilmente inserire stringhe di qualsiasi lunghezza, modificando il programma 11 in modo che il computer misuri la lunghezza della stringa che introduciamo e, in base a quella lunghezza, controlli il ciclo FOR...NEXT.
Le modifiche necessarie sono:

```
10 REM **TAGLIO DI STRINGA**
15 PRINT CHR$(147)
20 INPUT"INSERISCI UNA STRINGA";X$
30 PRINT TAB(10);"1234567890"
40 FOR I=1 TO LEN(X$)
50 LET A$=LEFT$(X$,I)
60 PRINT I;TAB(10);A$
70 NEXT I
80 END
```

LEN(X$) fa da limite superiore del ciclo

Programma 12

```
RUN
INSERISCI UNA STRINGA? STRINGARE
          1234567890
 1        S
 2        ST
 3        STR
 4        STRI
 5        STRIN
 6        STRING
 7        STRINGA
 8        STRINGAR
 9        STRINGARE

READY
```

- ESEGUI IL PROGRAMMA 12 -

Esercizio 3

Scrivi un programma che stampi nelle istruzioni DATA della risposta all'esercizio 1 le parole che cominciano per vocale.

Esercizio 4

Scrivi un programma per cambiare l'output dell'esecuzione RIGHT$ del Programma 11 in:

```
     F
    EF
   DEF
  CDEF
 BCDEF
ABCDEF
```

MID$(X$,I,J)

Abbiamo usato LEFT$ e RIGHT$ per tagliare l'estremità di una stringa ma potremmo voler tagliare una parte in mezzo ad una stringa, per es. M M in A A M M G G. Esiste un'altra istruzione BASIC;

MID$(X$,I,J)

che taglia una sottostringa di lunghezza J, a partire dalla posizione I:

```
MID$(X$, I, J)
         |  |
         |  |_____ lunghezza della sottostringa dalla posizione I
         |
         |________ posizione nella stringa
```

es. se X$ = POSIZIONE
MID$(X$,5,2) = ZI
e MID$(X$,2,4) = OSIZ

Useremo di nuovo il computer per vedere come funziona MID$ modificando ulteriormente il programma 11. Abbiamo già modificato il programma 11 in modo da poter immettere una stringa di lunghezza qualsiasi.

Perciò:

```
10 REM **STRINGHE**
15 PRINT CHR$(147)
20 INPUT"INSERISCI UNA STRINGA";X$
30 PRINT TAB(10);"1234567890"
40 FOR I=1 TO LEN(X$)
50 LET A$=LEFT$(X$,I)
60 PRINT;I;TAB(10);A$
70 NEXT I
80 END
```

Programma 13

Se ora cambiamo la linea 50 in:

```
50 LET A$=MID$(X$,I,1)
```

e immettiamo STRINGARE otteniamo:

```
RUN
INSERISCI UNA STRINGA? STRINGARE
          1234567890
 1        S
 2        T
 3        R
 4        I
 5        N
 6        G
 7        A
 8        R
 9        E

READY
```

Qui MID$ preleva tutte le possibili sottostringhe di lunghezza 1.
Se usiamo ora

```
50 LET A$=MID$(X$,I,2)
```

e immettiamo COSTRINGA otteniamo:

```
RUN
INSERISCI UNA STRINGA? COSTRINGA
          1234567890
 1        CO
 2        OS
 3        ST
 4        TR
 5        RI
 6        IN
 7        NG
 8        GA
 9        A ——————————————————— stringa "nulla"

READY
```

- ESEGUI IL PROGRAMMA 13 -

L'ultima sottostringa ci ha creato dei problemi. Non possiamo ottenere una sottostringa lunga 2 caratteri da una stringa di 9 caratteri a cominciare dal nono carattere. Se proviamo a farlo entriamo in uno stato di "default", e come risultato otteniamo una stringa nulla. Ci devono essere sempre caratteri sufficienti a sinistra della stringa originale per tirar fuori la sottostringa.
In generale, se vogliamo ottenere J caratteri, non possiamo cominciare a prelevare la sottostringa prima della (LEN(X$)–J+1) posizione.
Proprio cosi, +1!
Infatti, se LEN(X$)=10 e J=3, allora LEN(X$)–J=7;
ma possiamo ottenere una stringa di lunghezza 3 da una stringa di lunghezza 10 se partiamo da 8, e cioè utilizzando le posizioni 8, 9 e 10 della stringa originale.

DAV8

Scrivi un programma che accetti come input i numeri di telefono degli amici italiani di un nostro connazionale che vive a Parigi. Nella sua agenda sarà segnato il prefisso teleselettivo per l'Italia, il prefisso della città ed il numero telefonico.

In particolare

0039	81	738958
Italia	Pref. Napoli	Mario Rossi
0039	50	58773
Italia	Pref. Pisa	Luca Bianchi
0039	6	8979325
Italia	Pref. Roma	Aldo De Luca

E cioè, generalizzando, nella forma XXXX XXX XXXXXXX.
Fa in modo che il programma dia come risultato solo i prefissi, delle città (attenzione! ricorda che mentre in Italia devi comporre lo 0 prima del prefisso della città, dall'estero non devi farlo!) e ricorda che gli spazi sono caratteri proprio come le cifre.

Programmi con MID$

Come hai probabilmente capito, se vogliamo, MID$ può tagliare sottostringhe a sinistra e a destra. In altre parole, può darci ogni possibile sottostringa. Ecco un programma che: per prima cosa stampa tutte le sottostringhe di lunghezza 1, poi tutte quelle di lunghezza 2 e così via, finché stampa tutta la parola che è la sola sottostringa della stessa lunghezza della parola!

```
10 REM **ANCORA STRINGHE**
15 PRINT CHR$(147)
20 INPUT"INSERISCI UNA STRINGA";X$
30 PRINT"J";TAB(5);"β";TAB(10);"1234567890"
35 FOR J=1 TO LEN(X$)
40 FOR I=1 TO (LEN(X$)-J+1)
50 LET A$=MID$(X$,I,J)
60 PRINT J;TAB(5);I;TAB(10);A$
70 NEXT I
72 PRINT" "
75 NEXT J
80 END
```

J è la lunghezza della sottostringa che comincia da I

Programma 14

```
RUN
INSERISCI UNA STRINGA? ECCOLA

 J     I    1234567890
 1     1    E
 1     2    C
 1     3    C
 1     4    O
 1     5    L
 1     6    A

 2     1    EC
 2     2    CC
 2     3    CO
 2     4    OL
 2     5    LA

 3     1    ECC
 3     2    CCO
 3     3    COL
 3     4    OLA

 4     1    ECCO
 4     2    CCOL
 4     3    COLA

 5     1    ECCOL
 5     2    CCOLA

 6     1    ECCOLA

READY
```

(Nota le intestazioni e la scala)

- ESEGUI IL PROGRAMMA 14 -

5.7 VAL

Avendo trovato un metodo per tagliare una stringa, ora abbiamo bisogno di uno per esaminare quello che facciamo. Un metodo e quello di usare VAL(A) che legge il valore numerico di A.
VAL(A) ci da il valore numerico della stringa A, purche A cominci con =, o con una cifra. In tutti gli altri casi, VAL(A) > 0.

Programma per illustrare l'uso di VAL

Nel seguente programma immettiamo 7 stringhe (123456, 12345A,... AB-CDEF) e osserviamo il comportamento di VAL per le stringhe intere, per la sottostringa sinistra lunga 2 caratteri e per la sottostringa di mezzo, lunga 2 caratteri, a partire dal terzo.
Vedremo che se il carattere piu a sinistra nella stringa (o sottostringa) e una cifra, allora VAL ne stampa il valore anche se il resto della stringa contiene caratteri non numerici.

```
10 REM **LA FUNZIONE VAL**
15 PRINT CHR$(147)
17 REM **INIZIO DEL CICLO**
20 INPUT"PROSSIMA STRINGA";N$
25 IF N$="ZZZZ" THEN 990
30 LET N=VAL(N$)
40 LET P=VAL(LEFT$(N$,2))
50 LET Q=VAL(MID$(N$,3,2))
60 PRINT" "
70 PRINT N,P,Q
80 PRINT" "
90 GOTO 17
990 REM **FINE LAVORO**
999 END
```

Programma 15

```
RUN
PROSSIMA STRINGA? 123456

 123456       12          34

PROSSIMA STRINGA? 12345A

 12345        12          34

PROSSIMA STRINGA? 1234AB

 1234         12          34

PROSSIMA STRINGA? 123ABC

 123          12          3
```

VAL stampa il valore delle cifre a sinistra anche se la stringa contiene i caratteri non numerici A e B

```
PROSSIMA STRINGA? 12ABCD

 12          12          0
```

e ora in posizione 3, VAL(As)>0

```
PROSSIMA STRINGA? 1ABCDE

 1           1           0

PROSSIMA STRINGA? ABCDEF

 0           0           0

PROSSIMA STRINGA? ZZZZ

READY
```

- ESEGUI IL PROGRAMMA 15

DAV9

Quali sono i valori di:

(a) VAL(A) dove A > 54
(b) VAL(B) dove B > 76XY
(c) VAL(C) dove C > A3
(d) VAL(D) dove D > 132
(e) VAL(LEFT(E,2)) dove E > 593
(f) VAL(LEFT(F,1)) dove F > 8AM
(g) VAL(LEFT(G,2)) dove G > Z35
(h) VAL(RIGHT(H,1)) dove H > 593
(i) VAL(RIGHT(I,2)) dove I > 8AM
(j) VAL(RIGHT(J,2)) dove J > Z35

Controllo di stringhe di dati

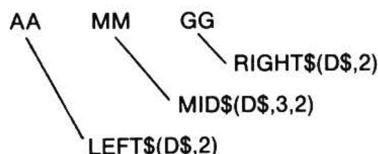
Vediamo ora come VAL puo essere usato in pratica, in questo caso per controllare l'esattezza dei dati battuti su un computer. E' questa una cosa tipica dei computer: poiche sappiamo che gli errori generalmente si verificano quando vengono battuti i dati, se e possibile, cerchiamo di usare il computer per rilevare tali errori.

Esercizio 5

Scrivi un programma che effettui il controllo dei dati sui 3 campi di una stringa di dati di 6 cifre.

Soluzione

La stringa dati D ha tre campi:



Abbiamo intenzione di considerare soltanto gli anni 19801981, cos*:
VAL(LEFT(D,2)) dovrebbe avere un'estensione di 8081,
VAL(MID(D,3,2)) dovrebbe avere un'estensione di 112,
VAL(RIGHT(D,2)) dovrebbe avere un'estensione di 131,

Questo e un processo abbastanza complesso per cui dobbiamo decidere quali passi seguire per arrivare ad una soluzione.
Vediamo come sono rappresentati tali passi nel diagramma di flusso di figura 3.

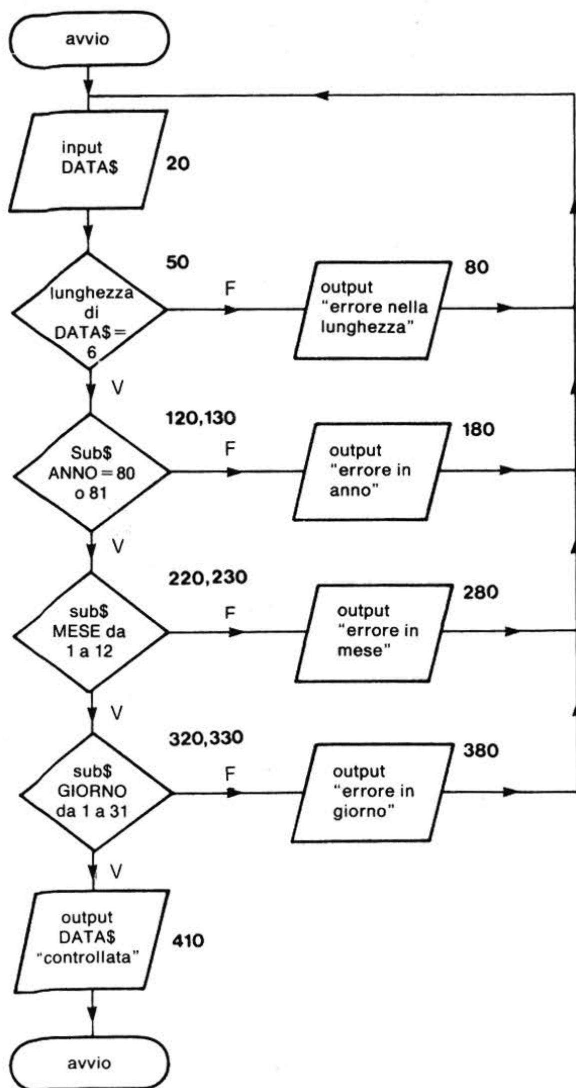


Figura 3 – Diagramma di flusso che illustra i 4 controlli su AAMMGG.

Il programma e molto simile in tutti e quattro i passi di controllo.
Se si verifica un errore, il programma lo segnala stampando un messaggio di errore e il controllo ritorna alla linea 20.

Se non ci sono errori, il programma gira fino alla linea 410 dove viene confermato che l'immissione e corretta.

```
10 REM **CONTROLLO DATA**
15 PRINT CHR$(147)
17 REM **INIZIO CONTROLLO**
20 INPUT"PROSSIMA DATA";D$
30 IF D$="ZZZZ" THEN 890
50 IF LEN(D$)=6 THEN 100 ]————————controllo lunghezza
80 PRINT"!!!ERRORE NELLA LUNGHEZZA DELLA DATA!!!"
90 GOTO 17
100 REM ********************
110 PRINT"LUNGHEZZA DELLA STRINGA: CORRETTA!"
120 IF VAL(LEFT$(D$,2))=80 THEN 200 ]— controllo anno
130 IF VAL(LEFT$(D$,2))=81 THEN 200 ]
180 PRINT"!!!!!!ERRORE NELL'ANNO!!!!!!"
190 GOTO 17
200 REM ********************
210 PRINT"ANNO: CORRETTO!"
220 IF VAL(MID$(D$,3,2))<1 THEN 270   ]- controllo mese
230 IF VAL(MID$(D$,3,2))<=12 THEN 300 ]
270 REM ********************
280 PRINT"!!!ERRRORE NEL MESE!!!"
290 GOTO 17
300 REM ********************
310 PRINT"MESE: CORRETTO!"
320 IF VAL(RIGHT$(D$,2))<1 THEN 370  ]-controllo giorno
330 IF VAL(RIGHT$(D$,2))<31 THEN 400 ]
370 REM ********************
380 PRINT"!!!ERRORE NEL GIORNO!!!"
390 GOTO 17
400 REM ********************
410 PRINT"LA DATA RISPETTA TUTTI I LIMITI"
490 GOTO 17
890 REM **FINE LAVORO**
900 PRINT"FINE DEL CONTROLLO SULLE DATE"
910 END
```

Programma 16

```
RUN
PROSSIMA DATA? 1234567
!!!ERRORE NELLA LUNGHEZZA DELLA DATA!!!
LUNGHEZZA DELLA STRINGA: CORRETTA!
!!!!!!ERRORE NELL'ANNO!!!!!!
PROSSIMA DATA? 803456
LUNGHEZZA DELLA STRINGA: CORRETTA!
ANNO: CORRETTO!
!!!ERRORE NEL MESE!!!
PROSSIMA DATA? 801256
LUNGHEZZA DELLA STRINGA: CORRETTA!
ANNO: CORRETTO!
MESE: CORRETTO!
!!!ERRORE NEL GIORNO!!!
PROSSIMA DATA? 800130
LUNGHEZZA DELLA STRINGA: CORRETTA!
ANNO: CORRETTO!
MESE: CORRETTO!
LA DATA RISPETTA TUTTI I LIMITI
PROSSIMA DATA? ZZZZ
FINE DEL CONTROLLO SULLE DATE

READY
```

- ESEGUI IL PROGRAMMA 16

5.8 Note su TAB: Caratteri di controllo

Alcuni BASIC hanno un'istruzione TAB del tipo:

PRINT TAB(X,Y) "—METTI QUI IL TUO MESSAGGIO"

dove X, numero di colonna dello schermo, e Y, numero di riga, individuano il punto da cui comincera la stampa del messaggio.
Questa e una possibilita utile per il BASIC offre per predisporre lo schermo facilmente con poche istruzioni.
Il BASIC del Commodore 64 non offre questo tipo di possibilita. Ma in alternativa si possono usare i comandi del cursore sopra-sotto-destra-sinistra con istruzioni di stampa letterale.

Per muovere il cursore di una riga in basso sullo schermo basta digitare PRINT seguito dal CRSR su-giù fra virgolette.
Per muovere il cursore di due righe in basso basta premere 2 volte il CRSR, e così via.
Per muovere il cursore in alto di una riga sullo schermo, identico procedimento: PRINT seguito da SHIFT + CRSR fra virgolette.
Per muovere il cursore di una colonna a destra PRINT seguito dall'altro CRSR (quello con le freccette destra - sinistra, questa volta) fra virgolette.
Per muoverlo a sinistra PRINT e SHIFT +CRSR sempre fra virgolette.
Infine, per posizionare il cursore nell'angolo in alto a sinistra dello schermo, PRINT + CLR HOME fra virgolette.
È possibile usare qualsiasi combinazione di questi caratteri di controllo all'interno delle virgolette di un'istruzione di stampa letterale. È logico comunque cominciare dall'angolo in alto a sinistra dello schermo spostandosi in giù di una riga per volta fino in fondo. Useremo i caratteri di controllo nelle unità del corso successive, ma e bene usarle limitatamente, perché possono rendere difficile da seguire il listing dei programmi, siccome vengono rappresentate con particolari caratteri grafici che voi stessi potrete individuare con la pratica.

Compito 5

1. Scrivi un programma che trovi la frequenza di ogni vocale nelle parole della filastrocca dell'esempio 2 e che dia anche un sommario del numero totale di vocali e consonanti di queste parole.
2. Scrivi un programma che ti permetta di inserire una stringa di caratteri e ottenere questa stringa in ordine inverso.

Obiettivi dell'unita 5

Controlla se ora sei capace di scrivere programmi semplici:
Usando LEN(A$) 
Usando LET C(I) > C(I) = 1 per contare le frequenze; 
Stampare un diagramma di frequenza 
Usando TAB per stampare in colonne 
Usando TAB per stampare una tabella di frequenza con le intestazioni 
Usando LEFT$(X$,I) 
e RIGHT$(X$,I) 
Usando MID$(X$,I,J) 
Usando VAL(A$) 

Risposte alle DAV e agli esercizi

DAV1

(a) 4; (b) 1; (c) 2 (non 72! Ricorda: LEN conta il numero dei caratteri); (d) 9 (LEN conta i caratteri senza tener conto se sono numeri, lettere o spazi).

DAV2

La tua tabella dovrebbe essere:

Lunghezza parole	numero	totale
1	1	4
2	~~1111~~	9
3	1111	1
4	~~1111~~ 1	5
5	~~1111~~ 1111	5
6		3
7	1	6
8	111	2
9	1	1
10		0
11		1
12	1	0
13		0
14		0
15		0

DAV3

(a) C(3)>1 (Non 3! C(3) ha contato quante volte, in tre immissioni, l'input corrispondeva al numero 3)

(b) C(9)>2
(c) C(1)>4
(d) C(0)>3

Esercizio 1

Una soluzione potrebbe essere quella che appare in risposta alla DAV4.

DAV4

```
**
*****
*******
********
***
*
```

alla fine apparirà il messaggio:
? OUT OF DATA ERROR in 10

DAV5

10 PRINT%COL 1+;TAB(9);%COL 2+;TAB(19);%COL 3+

DAV6

```
10  INPUT A,B,C
20  PRINT TAB(A);"TITOLO";TAB(B);"TITOLO";TAB(C),"TITOLO"
```

Programma 17

Esercizio 2

```
10 REM **DISTRIBUZIONE DI FREQUENZA**
15 REM **LA LISTA DELLE FREQUENZE E' F(K)**
20 PRINT CHR$(147)
25 DIM V$(5)
30 DIM F(6)
32 FOR I=1 TO 5     ]  legge le vocali dalla linea 800 e le
34 READ V$(I)       ]  memorizza in una lista
36 NEXT I
40 LET K=1                                          ]
45 REM **INIZIO DEL CICLO DI LETTURA**              ]  legge le
50 READ F(K)                                        ]  frequenze
60 IF F(K)=-9999 THEN 100                           ]  nella linea
70 LET K=K+1                                        ]  900
80 GOTO 45                                          ]
90 REM ********
100 REM **NON CONSIDERA IL NUMERO -9999**
110 LET N=K-1
```

```
200 REM **ROUTINE DI STAMPA**
210 PRINT
212 PRINT"VOCALE";TAB(9);"FREQ.";TAB(15)
    ;"GRAFICO"
214 PRINT"=========================="
220 FOR X=1 TO N
222 PRINT TAB(2);V$(X);TAB(7);"{SH B}";T
    AB(10);F(X);TAB(14);"{SH B}";TAB(16)
    ;
230 FOR Y=1 TO F(X)
240 PRINT"*";
250 NEXT Y
260 PRINT
280 NEXT X
290 PRINT"....SCALA....";TAB(15)"0....5..
    ..0....5....0"
300 REM ********
800 DATA A,E,I,O,U
900 DATA 9,16,10,10,6,-9999
```

Programma 18

```
RUN

VOCALE   FREQ. GRAFICO
==========================
  A        9    *********
  E        16   ****************
  I        10   **********
  O        10   **********
  U        6    ******
....SCALA....  0....5....0....5....0

READY
```

- ESEGUI IL PROGRAMMA 18 -

DAV7

(a) 1; (b) 1A2B; (c) C4D; (d) 3C4D.
Nota che LEFT e RIGHT trattano tutti i àcaratteri in una stringa nello stesso modo, non ha importanza se siano numeri o lettere.

Esercizio 3

```
10 REM **QUALI PAROLE INIZIANO CON UNA V
   OCALE?**
15 PRINT CHR$(147)
17 REM **CICLO DI LETTURA**
20 READ W$
30 IF W$="ZZZZ" THEN 9999
40 LET L$=LEFT$(W$,1)
50 IF L$="A" THEN 190
60 IF L$="E" THEN 190
70 IF L$="I" THEN 190
80 IF L$="O" THEN 190
90 IF L$="U" THEN 190
100 GOTO 17
190 REM *********
200 PRINT
210 PRINT L$,W$
220 GOTO 17
230 REM *********
900 DATA LA,PIGRIZIA,ANDO',AL,MERCATO,ED
    ,UN,CAVALLO,COMPRO'
910 DATA MEZZOGIORNO,ERA,SUONATO,QUANDO,
    A,CASA,RITORNO'
920 DATA PRESE,L'ACQUA,ACCESE,IL,FUOCO,S
    I,SEDETTE,E,RIPOSO'
930 DATA ED,INTANTO,A,POCO,A,POCO
940 DATA ANCHE,IL,SOLE,TRAMONTO',ZZZZ
9990 REM **FINE LAVORO**
9999 END
```

Programma 19

```
RUN

A          ANDO'

A          AL

E          ED

U          UN

E          ERA
```

```
A          A

A          ACCESE

I          IL

E          E

E          ED

I          INTANTO

A          A

A          A

A          ANCHE

I          IL

READY
```

- ESEGUI IL PROGRAMMA 19 -

Esercizio 4

```
10 REM **STRINGHE**
15 PRINT CHR$(147)
20 INPUT"INSERISCI UNA STRINGA DI 6 CARATTERI ";X$
30 PRINT TAB(10);"1234567890"
40 FOR I=1 TO 6
50 LET A$=RIGHT$(X$,I)
60 PRINT;I;TAB(16-1);A$
70 NE:'T I
80 END
```

Programma 20

```
RUN
INSERISCI UNA STRINGA DI 6 CARATTERI? AB
CDEF
           1234567890
 1              F
 2              EF
 3              DEF
 4              CDEF
 5              BCDEF
 6              ABCDEF

READY
```

- ESEGUI IL PROGRAMMA 20 -

DAV8

```
5 PRINT CHR$(147)
10 INPUT"PROSSIMO NUMERO DI TELEFONO";N$
20 LET A$=MID$(N$,5,3)
30 PRINT A$
40 GOTO 10
```

Programma 21

- ESEGUI IL PROGRAMMA 21 -

DAV9

(a) 54; (b) 76 (si ferma quando trova le lettere); (c) 0 (comincia con una lettera, perciò zero 0);
(d) –132; (e) 59; (f) 8; (g) 0 (comincia con una lettera); (h) 3; (i) 0 (comincia da A che è una lettera); (j) 35.

UNITÀ 6
Parliamo un po' di dadi e di giochi

6.1 Numeri casuali

La funzione di programmazione che ci permette di dare un pizzico di stranezza ad un programma è quella che genera numeri casuali. Questa funzione è il cuore di molti programmi di giochi e di simulazione oggi disponibili su ogni microcomputer.
Ti sarai imbattuto nei numeri casuali facendo giochi come il lancio di una moneta o di un dado, o estraendo numeri da un cappello.
Questi giochi così comuni e familiari sono stati "istituzionalizzati", con le slot machine, nelle sale da gioco.
Sebbene tutti noi abbiamo un'idea intuitiva di cosa si intenda con sequenza di numeri casuali, è abbastanza difficile definire chiaramente questo concetto. Diamo un'occhiata a qualche sequenza di numeri per provare a chiarirci le idee.

Abbiamo qui tre esperimenti immaginari, ognuno dei quali consiste nel lancio di dadi a sei facce, per quindici volte. Immaginiamo che nel primo esperimento i risultati siano stati quelli indicati nella sequenza A della figura 1. Il secondo esperimento ha generato i numeri illustrati nella sequenza B ed il terzo i numeri della sequenza C.

Sequenza A
5, 1, 2, 4, 6, 3, 2, 1, 6, 3, 5, 4, 3, 4, 2

Sequenza B
6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6

Sequenza C
1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 5, 6, 1, 2, 3,

Figura 1 – Sequenze di numeri casuali?

Molti di noi sarebbero propensi a credere che la sequenza A rappresenti una sequenza tipica di numeri generati lanciando un dado per quindici volte. Infatti questa sequenza di numeri non mostra schemi definibili o ripetizioni e l'occorrenza di ogni numero sembrerebbe essere ugualmente probabile. Non siamo sorpresi dall'aspetto di una delle sottosequenze in questa sequenza principale.

Per contrasto, invece, la sequenza B è abbastanza irragionevole. Certamente non ci aspetteremmo di ottenere quindici sei in quindici lanci consecutivi del

dado. Anzi, penseremmo sicuramente di avere davanti a noi un dado truccato. Intuitivamente, mentre accettiamo il fatto che la sequenza A è frutto di lanci a caso, non pensiamo la stessa cosa per la sequenza B.
Un'altra caratteristica delle sequenze di numeri casuali, che abbiamo appreso dall'esperienza, è che, su lunghe sequenze, possono verificarsi degli strani risultati localizzati. Quello che vogliamo dire è che dopo, diciamo, un centinaio di lanci, ci aspetteremo in media circa sedici uno, circa sedici due, sedici tre e così via. In altre parole, su sequenze più lunghe, ci aspettiamo che si applichino le leggi del caso. Se consideriamo ora la sequenza C con il suo schema ...6,1,2,3,4,5,6,1... continuato per un centinaio di lanci, allora l'effetto a lungo termine di emergenza di valori medi sarebbe soddisfatto. Ma, ancora una volta, questa sequenza non sarebbe accettabile intuitivamente come casuale perché non ci aspettiamo che questo schema sequenziale persista per un centinaio di lanci solo per caso. Questi concetti di media statistica su una frequenza di lanci e di "ragionevolezza" dello schema dei numeri in sequenza vengono acquisiti intuitivamente nel gioco delle probabilità. Ci sono tecniche statistiche per controllare queste due caratteristiche di sequenza di numeri a caso, ma non ci occuperemo qui di tali tecniche.

Un computer è una macchina molto rigida, perciò ti sorprenderà sapere che nella macchina devono essere programmate delle caratteristiche abbastanza speciali per ottenere una sequenza di numeri casuali. Per i nostri usi, comunque, assumeremo che la tabella dei numeri casuali sia stata immagazzinata nella memoria della macchina.
La sequenza dei numeri è molto lunga e dovrebbe essere generata moltissime volte perché appaia ripetuta.
Per ottenere una sequenza differente di numeri casuali da un'esecuzione di programma ad un altro, tutto ciò che una macchina deve fare è cominciare a leggere questa tabella di numeri casuali da un punto differente.
Il punto iniziale spesso viene chiamato "seme" e tratteremo le sequenze di numeri casuali partendo da semi differenti.
Dal momento che il computer deve "escogitare" sequenze di numeri casuali, i numeri prodotti sono chiamati generalmente "numeri pseudocasuali".

6.2 La funzione RND

RND è una funzione che richiede un argomento, deve cioè essere seguita da un numero tra parentesi:

RND(A)

L'argomento "A" di RND può essere negativo, positivo o uguale a zero ed il programma che segue illustra gli effetti per differenti valori di A.

```
10 REM **LA FUNZIONE RND**
15 PRINT CHR$(147)
20 INPUT"ARGOMENTO DI RND";A
30 FOR I=1 TO 10
40 LET B=RND(A)
50 PRINT B
60 NEXT I
70 END
```

Programma 1 - Effetto di A su RND.

```
RUN
ARGOMENTO DI RND? -1
 2.99196472E-08
 2.99196472E-08
 2.99196472E-08
 2.99196472E-08
 2.99196472E-08
 2.99196472E-08
 2.99196472E-08
 2.99196472E-08
 2.99196472E-08
 2.99196472E-08

READY
```

Effetti di A negativo sul numero casuale

```
RUN
ARGOMENTO DI RND? 0
 .253907502
 .527345121
 .8007828
 .539064288
 .089845717
 .492189646
 .316408694
 .914065182
 .308596611
 .48047173
```

Anche A posto a 0 genera numeri casuali

```
RUN
ARGOMENTO DI RND? 1
 .186647695
 .531850191
 .615078971
 .862640194
 .868612207
 .665991999
 .168051895
 .749537644
 .921491476
 .376282314

READY
```

con A=1 i numeri sembrano davvero casuali

- ESEGUI IL PROGRAMMA 1 -

Per vedere l'effetto di A sul tuo microcomputer.
Per ciò che concerne la generazione di numeri casuali, ignoreremo l'argomento diverso da 1.

RND(1)

La ricerca che abbiamo fatto sul nostro calcolatore dimostra che

RND(1)

darà numeri casuali all'interno dello spazio 0–1. Cambiare l'argomento A non sembra una soluzione efficace per estendere l'ampiezza dello spazio. Ed allora, come possiamo ottenere altri numeri casuali? In modo abbastanza semplice: moltiplichiamo RND(1) per un altro numero. Perciò

RND(1) da un numero casuale nello spazio 0–1;
6*RND(1) da un numero casuale nello spazio 0–6;
e 52*RND(1) da un numero casuale nello spazio 0–52, etc.

Puoi pensare a RND come ad un "fattore di conversione" che cambia a tuo piacimento. Il programma seguente tenta di chiarire questo concetto.

```
10 REM *RND CON FATTORE DI CONVERSIONE*
20 PRINT CHR$(147)
30 PRINT"β";TAB(9);"RND(1)";TAB(22);"6*RND(1)"
```

```
40 PRINT"--------------------------------------------"
50 FOR I=1 TO 10
60 LET B=RND(1)
70 LET C=6*B
80 PRINT I;TAB(9);B;TAB(22);C
90 NEXT I
100 END
```

Programma 2 - RND(1) fattore di conversione.

```
RUN
 1          RND(1)          6*RND(1)
---------------------------------------------
 1          .212737779      1.27642667
 2          .906009651      5.43605791
 3          .319934526      1.91960715
 4          .658507938      3.95104763
 5          .669658419      4.01795051
 6          .973812517      5.84287511
 7          .594765049      .856859029
 8          .509851446      3.05910868
 9          .421018047      2.52610829
 10         .520486986      3.12292192

READY
```

Esegui la linea 70 con C=52*B, cambiando l'intestazione sulla linea 30

```
10 REM *RND CON FATTORE DI CONVERSIONE*
20 PRINT CHR$(147)
30 PRINT"1";TAB(9);"RND(1)";TAB(22);"52*RND(1)
40 PRINT"--------------------------------------------"
50 FOR I=1 TO 10
60 LET B=RND(1)
70 LET C=52*B
80 PRINT I;TAB(9);B;TAB(22);C
90 NEXT I
100 END
```

Programma 3 - RND(1) fattore di conversione.

```
RUN
 1            RND(1)             52*RND(1)
--------------------------------------------
 1            .841374074         43.7514519
 2            .0176775215        .919231117
 3            .485276508         25.2343784
 4            .207060736         10.7671583
 5            .129610216         6.73973123
 6            .175612672         9.13185895
 7            .534037232         27.7699361
 8            .728265509         37.8698065
 9            .620224149         32.2516558
 10           .622480087         32.3689645

READY
```

- ESEGUI IL PROGRAMMA 3 -

DAV1

Scrivi un programma per stampare sei numeri casuali tra 0 e 5.999999.

La funzione RND + 1

Se guardi di nuovo l'output del programma 2, nell'esecuzione con 6*RND(1), i numeri erano:

```
 1.27642667
 5.43605791
 1.91960715
 3.95104763
 4.01795051
 5.84287511
 .856859029
 3.05910868
 2.52610829
 3.12292192

READY
```

Guarda ora i numeri prima del punto decimale.
Sono:

1,5,1,3,4,5,0,3,2,3

e cioé i membri dell'insieme

(0,1,2,3,4,5).

Ma, se stessimo lanciando un dado, otterremmo appunto numeri membri dell'insieme (1,2,3,4,5,6). Tutto ciò che dobbiamo fare allora è aggiungere 1 ad ogni membro del primo insieme per ottenere il secondo. Ora, giocando, ci capita di frequente di voler lanciare un dado (risultati 1, 2, 3, 4, 5, 6) o di usare delle carte da poker (52 risultati), perciò siamo interessati in modo particolare alle funzioni.

6*RND(1)+1 e 52*RND(1)+1

Il programma seguente ci permette di esplorare tali funzioni

```
10 REM *RND CON FATTORE DI CONVERSIONE*
20 PRINT CHR$(147)
30 PRINT"1";TAB(9);"RND(1)";TAB(22);"6*RND(1)+1"
40 PRINT"------------------------------------'
50 FOR I=1 TO 10
60 LET B=RND(1)
70 LET C=6*B+1
90 PRINT I;TAB(9);B;TAB(22);C
100 NEXT I
110 END
```

Programma 4 - 6*RND(1)+1

```
RUN
 1          RND(1)          6*RND(1)+1
------------------------------------
 1          .123215189      1.73929113
 2          .0347393571     1.20843614
 3          .296775924      2.78065555
 4          .984072616      6.9044457
 5          .911611658      6.46966995
 6          .628360668      4.77016401
 7          .912091129      6.47254678
 8          .527776101      4.16665661
 9          .942754194      6.65652517
 10         .812559406      5.87535644

READY
```

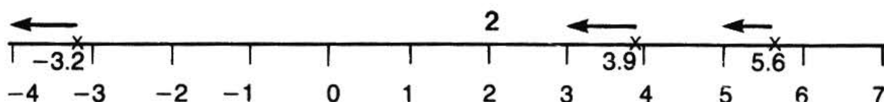
- ESEGUI IL PROGRAMMA 4 -

La funzione INT

Se ora guardi le cifre a sinistra del punto nella terza colonna delle esecuzioni del programma 4, vedrai che abbiamo generato i numeri casuali che volevamo. La colonna 3 contiene numeri da 1 a 6.
Ma cosa ce ne facciamo di tutto quello che si trova a destra del punto decimale? Beh, possiamo sbarazzarcene utilizzando la funzione INT. L'effetto di INT(X) è quello di darci il numero intero X, o la parte intera di X che sia l'intero più grande ma non maggiore di X. L'effetto di INT è quello di arrotondare sempre per difetto al precedente numero intero:

INT (5.6) = 5
INT (3.9 = 3
INT(–3.2) = –4
INT(2) = 2

Se il risultato INT (–3.2) = –4 ti sorprende, guarda la rappresentazione numerica sulla retta e ricorda che INT arrotonda sempre per difetto all'intero precedente. Attento! non arrotonda MAI per eccesso i numeri.



DAV2

Quali sono i valori di:
(a) INT(4.5)
(b) INT(9.1)
(c) INT(–2.5)
(d) INT(–0.99)
(e) INT(1.01)
Ora, con INT, possiamo generare i numeri interi da 1 a 6 e da 1 a 52 per usarli come dadi o carte da poker. Quello di cui abbiamo bisogno è

INT(6*RND(1) + 1)
INT(52*RND(1) + 1)

Il programma seguente stampa i valori di queste due funzioni:

```
10 REM **LA FUNZIONE INT**
20 PRINT CHR$(147)
30 PRINT"  1"TAB(6);"RND(1)";TAB(17);"INT
(6*B+1)";" INT(52*B+1)"
40 PRINT"¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
¬¬¬¬¬¬¬¬¬"
50 FOR I=1 TO 10
60 LET B=RND(1)
70 LET C=INT(6*B+1)
80 LET D=INT(52*B+1)
90 PRINT I;TAB(5);B;TAB(21);C;"          ";D
100 NEXT I
110 END
JREADY.
J
```

Programma 5 - INT per ottenere numeri interi.

```
RUN
 1     RND(1)       INT(6*B+1) INT(52*B+1)
¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
 1     .714062732       5           38
 2     .0730809871      1           4
 3     .17016535        2           9
 4     .786998985       5           41
 5     .645609412       4           34
 6     .419266636       3           22
 7     .580856299       4           31
 8     .248696615       2           13
 9     .494313401       3           26
 10    .576983732       4           31

READY
```

- ESEGUI IL PROGRAMMA 5 -

6.3 Post scriptum sui numeri casuali

Il seguente programma simula il lancio di un dado per 100 volte.

```
10 REM **LANCIO DADO PER 100 VOLTE**
20 PRINT CHR$(147)
25 PRINT"LANCIO DI UN DADO"
30 FOR I=1 TO 10
```

```
40 FOR J=1 TO 10
50 LET X=INT(6*RND(1))
60 PRINT X;
70 NEXT J
80 PRINT
90 NEXT I
100 END
```

Programma 6 - Lancio di un dado.

```
RUN
LANCIO DI UN DADO

 2 3 2 3 2 5 4 0 4 4
 2 2 5 3 4 1 0 4 0 5
 1 5 0 2 5 4 5 3 3 1
 4 3 2 4 5 3 4 1 3 1
 3 0 1 4 0 3 2 5 0 2
 4 5 4 4 4 4 2 3 3 0
 1 1 3 0 5 2 4 0 2 2
 3 0 5 3 3 0 2 3 3 1
 5 5 2 4 4 0 5 5 2 3
 3 1 2 1 1 1 4 4 1 0

READY
```

- ESEGUI IL PROGRAMMA 6 -

Per essere sicuro di aver capito la funzione INT, prova a rispondere alle seguenti domande:

DAV3

Il programma:

```
15 PRINT CHR$(147)
20 FOR X=-3.8 TO -1.8 STEP(.2)
30 LET Y=INT(X)
40 PRINT X,Y
50 NEXT X
60 END
```

Programma 7

stampa 10 paia di numeri. Quali sono?

DAV4

Il programma

```
10 REM **DAV4**
15 PRINT CHR$(147)
20 FOR X=1.6 TO 3.4 STEP(.2)
30 LET Y=INT(X)
40 PRINT X,Y
50 NEXT X
60 END
```

Programma 8

stampa 9 paia di numeri. Quali sono?

6.4 Due esempi

Questo paragrafo contiene due esempi. Ti suggeriamo di affrontarli prima come se fossero esercizi e di confrontare la tua soluzione con la nostra.

Esempio 1

Scrivi un programma che simuli il lancio di una moneta per 100 volte. Conta e stampa il numero di volte che la moneta cadrà con la faccia raffigurante testa o con la faccia raffigurante la croce.

Soluzione

Il cuore della soluzione sta in un generatore di numeri casuali che produce un 1 o un 2.

Useremo 1 per rappresentare la croce e 2 per rappresentare la testa. Usando questo approccio, l'algoritmo descrittivo per la soluzione del problema è:

1. Inizio
2. Poni il totale delle teste e delle croci a 0
3. Inserisci il contatore di cicli

4. Genera casualmente i due valori 1 e 2
5. Se il numero casuale è uguale a 2 allora vai all'istruzione 8 altrimenti vai alla 6.
6. Aggiungi 1 al totale delle croci
7. Vai all'istruzione 9
8. Aggiungi 1 al totale delle teste
9. Aggiungi 1 al contatore di cicli
10. Se il contatore di cicli è < = 100 allora vai all'istruzione 4 altrimenti vai all'istruzione 11
11. Emetti il totale di teste e croci
12. Stop

Figura 2 – Soluzione descrittiva del lancio della moneta.

In alternativa, puoi preferire una soluzione di descrizione su un diagramma di flusso:

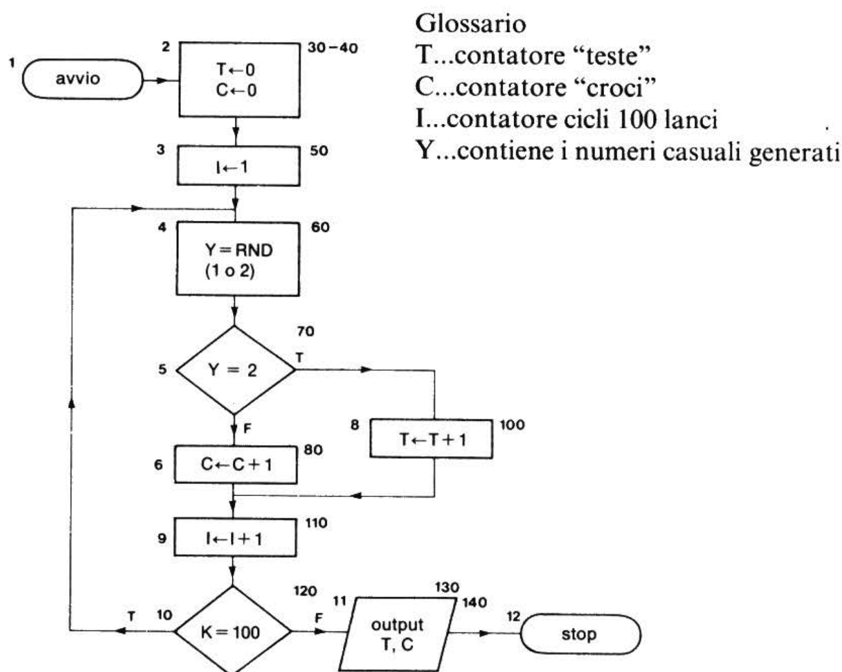


Figura 3 – Diagramma di flusso per il lancio della moneta.

e, finalmente, il programma

```
10 REM **LANCIO MONETA PER 100 VOLTE**
20 PRINT CHR$(147)
25 PRINT"RISULTATO LANCIO DELLA"
27 PRINT"MONETA PER 100 VOLTE"
30 LET T=0
40 LET C=0
50 LET I=1
55 REM **LANCIO DELLA MONETA**
60 LET Y=INT(2*RND(1)+1)
70 IF Y=2 THEN 95
80 LET C=C+1
90 GOTO 105
95 REM **2=TESTA**
100 LET T=T+1
105 REM **CONTEGGIO TOTALE**
110 LET I=I+1
120 IF I<=100 THEN 55
125 PRINT
130 PRINT"TESTE","CROCI"
140 PRINT T,C
150 END
```

Emissioni tipiche

```
RUN
RISULTATO LANCIO DELLA
MONETA PER 100 VOLTE

TESTE       CROCI
 51          49

READY

RUN
RISULTATO LANCIO DELLA
MONETA PER 100 VOLTE

TESTE       CROCI
 57          43

READY
```

```
RUN
RISULTATO LANCIO DELLA
MONETA PER 100 VOLTE

TESTE       CROCI
 55          45

READY
```

Programma 9 * Lancio di una moneta.

- ESEGUI IL PROGRAMMA 9 -

Esempio 2

Scrivi un programma di simulazione del lancio di 2 monete per 100 volte. Conta ed emetti il numero di croci. I risultati di questo esperimento immaginario sono: testa–testa; croce–croce; testa–croce; croce–testa.

Soluzione

Useremo la stessa regola di attribuire il punteggio 1 per la croce e 2 per la testa, ma ricordiamoci che stiamo lanciando 2 monete.
immagazziniamo il punto a favore della prima moneta in C1 e della seconda in B2. Quindi sommiamo C1 e B2 per avere il punteggio totale del lancio:

S = C1 + B2

S può essere 2, 3 o 4:

lancio	punteggio
CC	1+1=2
CT o TC	1+2=2+1=3
TT	2+2=4

Poi contiamo quanti 2, 3 e 4 otteniamo.

Glossario di simboli
T2...totale di TT
C2...totale di CC
M1...totale di TC o CT
C1...1 o 2 casuale per la moneta 1
B2...1 o 2 casuale per la moneta 2
S...somma dei punteggi.

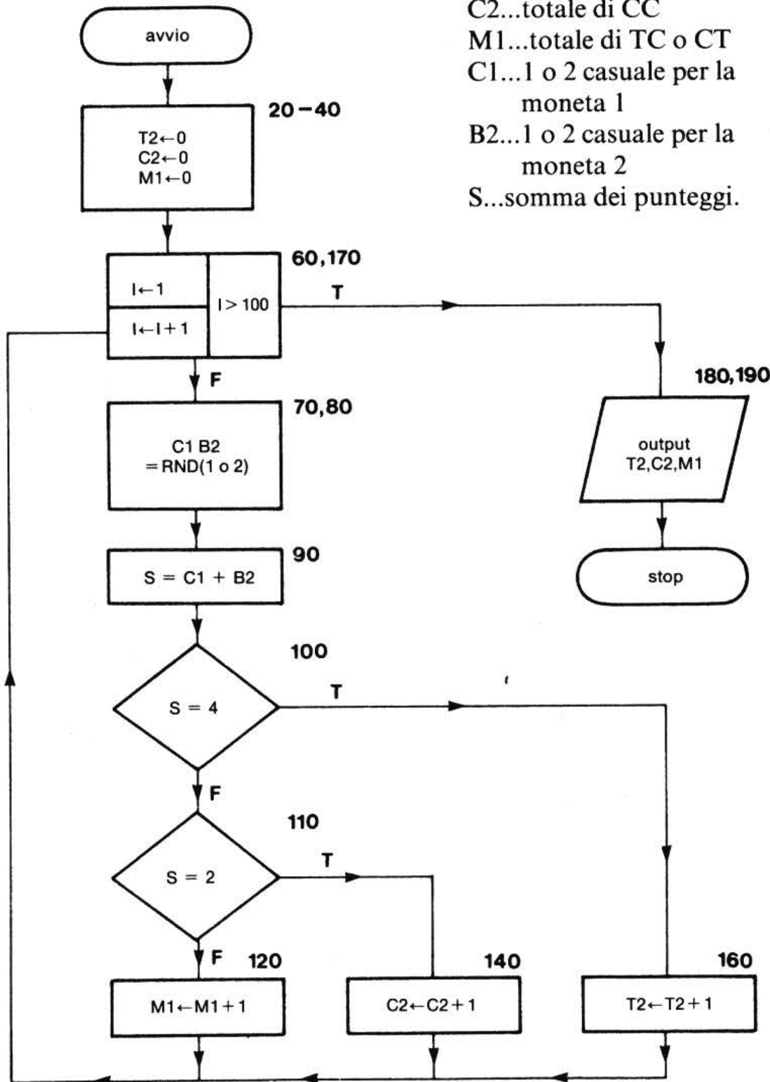


Figura 4 – Diagramma di flusso per il lancio di due monete.

Contatore dei 2 C2
Contatore dei 3 M1 (Per testa(T) e croce(C) insieme)
Contatore dei 4 T2

Il diagramma di flusso della soluzione, in figura 4, genera il seguente programma:

```
10 REM **LANCIA 2 MONETE 100 VOLTE**
15 PRINT CHR$(147)
17 PRINT"LANCIA 2 MONETE 100 VOLTE"
20 LET T2=0
30 LET C2=0
40 LET M1=0
50 REM **CICLO DI LANCIO**
60 FOR I=1 TO 100
70 LET C1=INT(2*RND(1)+1)
80 LET B2=INT(2*RND(1)+1)
90 LET S=C1+B2
100 IF S=4 THEN 155
110 IF S=2 THEN 135
120 LET M1=M1+1
130 GOTO 165
135 REM **DUE CROCI**
140 LET C2=C2+1
150 GOTO 165
155 REM **DUE TESTE**
160 LET T2=T2+1
165 REM **RITORNO AL LANCIO SUCCESSIVO**
170 NEXT I
180 PRINT"CC","TC","TT"
190 PRINT C2,M1,T2
200 END
```

Programma 10 - Lancio di due monete

```
RUN
LANCIA 2 MONETE 100 VOLTE

 CC          TC          TT
 26          50          24

READY
```

```
RUN
LANCIA 2 MONETE 100 VOLTE

 CC            TC            TT
 28            44            28

READY


RUN
LANCIA 2 MONETE 100 VOLTE

 CC            TC            TT
 24            47            29

READY


RUN
LANCIA 2 MONETE 100 VOLTE

 CC            TC            TT
 21            57            22

READY
```

- ESEGUI IL PROGRAMMA 10 -

6.5 Dove conservare i punteggi

Avrai notato che abbiamo usato qualche nome di variabile poco chiaro, come C2, T2 e M1.
Forse hai pensato "e le liste? Non potrebbero esserci utili qui come se fossero tabelle di frequenza?". Effettivamente potrebbero esserci utili, perciò proviamo ad usare una lista di punteggi S(I) nel lancio di una moneta. Diciamo allora:
Se il punteggio è 2, aggiungi 1 al numero in S(2)
Se il punteggio è 3, aggiungi 1 al numero in S(3)
Se il punteggio è 4, aggiungi 1 al numero in S(4)
e in generale
se il punteggio è S, aggiungi 1 al numero in S(S)

Applicazioni al lancio di due monete

Se torniamo all'esempio 2 possiamo utilizzare le linee da 10 a 90 e poi metterle nel nostro nuovo sistema di punteggio:

```
100 LET S(S)=S(S)+1
```

Il programma allora diventa:

```
10 REM **LANCIA 2 MONETE 100 VOLTE**
15 PRINT CHR$(147)
17 PRINT"LANCIA 2 MONETE 100 VOLTE"
18 DIM S(4)
20 LET S(4)=0
30 LET S(3)=0
40 LET S(2)=0
50 REM **CICLO DI LANCIO**
60 FOR I=1 TO 100
70 LET C1=INT(2*RND(1)+1)
80 LET B2=INT(2*RND(1)+1)
90 LET S=C1+B2
100 LET S(S)=S(S)+1
110 NEXT I
120 PRINT"CC","TC","TT"
130 PRINT S(2),S(3),S(4)
140 END
```

Programma 11

```
RUN
LANCIA 2 MONETE 100 VOLTE

 CC          TC          TT
 27          44          29

READY


RUN
LANCIA 2 MONETE 100 VOLTE

 CC          TC          TT
 23          56          21

READY
```

```
RUN
LANCIA 2 MONETE 100 VOLTE

 CC           TC           TT
 18           58           24

READY


RUN
LANCIA 2 MONETE 100 VOLTE
 CC           TC           TT
 20           49           31

READY
```

- ESEGUI IL PROGRAMMA 11 -

Lista di punteggi nel lancio di un dado

La lista di punteggi nel lancio di un dado sarebbe:

S(1), S(2), S(3)...S(6);

e per il lancio di due dadi:

S(2), S(3), S(4)...S(12).

Esercizio 1

Scrivi un programma per simulare il lancio di un dado per 100 volte. Conta e stampa il numero di volte in cui si verifica ciascun punteggio.

Esercizio 2

Modifica il programma scritto per l'esercizio 1 in modo da simulare il lancio di due dadi per 100 volte.

Esercizio 3

Scrivi un programma in modo che sullo schermo appaiano i dati ottenuti nel programa dell'esercizio 2, in forma di diagramma di frequenza.

6.6 Brevi tagli nella scrittura del programma

I nostri programmi cominciano ad essere troppo lunghi e più lunghi essi sono, più tempo si impiega a batterli sul computer. E' possibile operare allora dei brevi tagli che rendono più veloce l'immissione. Finora abbiamo evitato di fare tagli nel programma perché crediamo che la chiarezza sia molto più importante della velocità. Cercheremo di fare qualche abbreviazione all'interno del programma per mostrarti come si procede ma continueremo la nostra politica di preferire la chiarezza di interpretazione.

I principali tipi di tagli sono:

1. La parola LET può essere omessa nelle istruzioni di assegnazione. Quindi: 20 LET A = B può essere scritta 20 A = B.

2. E' permesso scrivere più di un'istruzione per linea; le istruzioni devono essere separate dal segno di due punti, perciò:

```
10 LET A=7
20 LET B=8
30 PRINT A+B
```

può essere scritto:

```
10 LET A=B: LET B=8: PRINT A+B
```

o, usando anche il primo taglio:

```
10 A=7: B=8: PRINT A+B
```

(Questo taglio rende più veloce l'esecuzione del programma e il tempo di battitura. I computer perdono del tempo per interpretare ogni numero di linea, così meno sono i numeri di linea, più veloce sarà l'esecuzione).

3. La parola PRINT può essere sostituita dal punto interrogativo (?). Comunque quando tu chiedi al computer di visualizzare il programma, vedrai che il "?" è sostituito dalla parola PRINT.

   E cioè se batti

   10 A=7: B=8:? A+B

   e dai il comando LIST, otterrai:

   10 A=7: B=8: PRINT A+8

   L'appendice D del Manuale dell'utente elenca tutte le possibili abbreviazioni delle parole chiave.

4. Puoi includere anche delle espressioni aritmetiche nell'istruzione PRINT. Tali espressioni verranno eseguite. E' quello che abbiamo già fatto quando abbiamo scritto PRINT A+B.

Ma attenzione,

- non riempire di istruzioni una linea solo per il gusto di farlo;

- presta attenzione al trasferimento delle istruzioni di controllo, per es. GOTO..., IF..., THEN..., e più tardi GO SUB... Ricorda, le linee a cui ci riferiamo devono esistere e il controllo di programma va all'inizio della linea nell'istruzione GOTO;

- molti programmi impiegano l'80% del tempo nell'esecuzione e il 20% nella codifica. Perciò, concentra le abbreviazioni (e quindi la velocità) sulle istruzioni più complesse, per es. le linee 70-120 nella soluzione all'esercizio 3;

- il numero di linea associato con un'istruzione REM fa perdere del tempo al computer, perciò sarebbe opportuno aggiungere REM... alla fine di altre istruzioni.
  Lo faremo nei programmi successivi. Non pensare che però sia tutto così chiaro.

- distingui chiaramente fra i due punti (:) e il punto e virgola (;) nel listing dei programmi tuoi e nostri e specialmente in quelli nelle riviste di computer. Le coppie (, < e > ,) sono spesso scarsamente distinguibili.

- Nell'unità didattica 10 il comando PRINT deve essere battuto completamente.? # non è accettabile dal Commodore 64.

Alcuni esempi di tagli brevi

Il programma 11, che aveva 14 linee, può essere riscritto in 10 linee

```
10 REM **LANCIA 2 MONETE 100 VOLTE**
15 PRINT CHR$(147)
20 DIM S(4):S(4)=0:S(3)=0:S(2)=0
50 PRINT"LANCIO DI DUE MONETE"
60 FOR I=1 TO 100
70 C1=INT(2*RND(1)+1):B2=INT(2*RND(1)+1)
90 S=C1+B2:S(S)=S(S)+1
110 NEXT I
120 PRINT"CC","TC","TT"
130 PRINT S(2),S(3),S(4)
140 END
```

3 istruzioni di assegnazione senza LET (linea 20)

Qui cominciano i lanci.

Le linee da 70 a 90 sono eseguite 100 volte.

Programma 12

DAV5

Prova a fare qualche breve taglio sul Programma 20 (soluzione dell'esercizio 3).

6.7 Concatenazione

Dopo aver incontrato un mucchio di problemi nell'Unità 5 per tagliare le stringhe, ora invece parleremo di come rimetterle insieme.
Il Programma 13 ti mostra cosa succede.

```
10 REM **CONCATENAZIONE**
15 PRINT CHR$(147)
20 INPUT"PRIMA STRINGA";A$
30 INPUT"SECONDA STRINGA";B$
40 PRINT A$+B$
50 END
```

Programma 13 - Concatenazione

```
RUN
PRIMA STRINGA? METTI
SECONDA STRINGA? INSIEME
METTIINSIEME

READY
```

```
RUN
PRIMA STRINGA? CONCATE
SECONDA STRINGA? NAZIONE
CONCATENAZIONE

READY
```

DAV6

Cogliamo l'occasione per fare un po' di inglese. Come forse saprai, nella lingua inglese il plurale di una parola generalmente si forma aggiungendo una *s* al singolare.
Consulta un qualsiasi dizionario di lingua inglese tuo o di un tuo amico e scegli qualche nome. Scrivi un programma per immettere una parola inglese ed emettere il plurale, assumendo che tutte le parole abbiano bisogno solo della s per diventare plurali.
Il programma 14 mostra come possiamo usare la concatenazione per costruire una stringa da una lista di simboli. Abbiamo immagazzinato le lettere sulla linea 40(DATA) e nel ciclo 110–140 abbiamo aggiunto una nuova lettera alla stringa per ogni passaggio del ciclo.

```
10 REM **ANCORA SULLE CONCATENAZIONI**
20 REM **PREPARA UN ELENCO**
22 PRINT CHR$(147)
25 DIM A$(10)
30 FOR I=1 TO 10:READ A$(I):NEXT I
40 DATA A,B,C,D,E,F,G,H,I,J
50 REM **********
100 C$="":REM **VUOTA C$**
110 FOR J=1 TO 10
120 C$=C$+A$(J)
130 PRINT J,C$
140 NEXT J
150 END
```

Programma 14

```
RUN
 1          A
 2          AB
 3          ABC
 4          ABCD
 5          ABCDE
 6          ABCDEF
 7          ABCDEFG
 8          ABCDEFGH
 9          ABCDEFGHI
 10         ABCDEFGHIJ

READY
```

- ESEGUI IL PROGRAMMA 14 -

Questo processo ha una grossa importanza nell'analisi dei testi, ma noi lo useremo solo per codici e giochi.

6.8 STR$

Questa funzione ha l'effetto contrario alla funzione VAL. Mentre la funzione VAL ci dà il valore numerico di una stringa, la funzione STR$ cambia un numero in una stringa di caratteri.
STR$(X) dà la rappresentazione del valore di X in forma di stringa.

Stampa di STR$

STR$(N) sembra molto simile ad N stesso, come illustra il seguente programma:

```
10 REM **LA FUNZIONE STR$**
15 PRINT CHR$(147)
20 INPUT"DAMMI UN NUMERO";N
25 PRINT"01234567890123456789 0"
30 PRINT N,STR$(N)
40 END
```

Programma 15

```
RUN
DAMMI UN NUMERO? 17
01234567890123456789O
 17          17

READY
```

```
RUN
DAMMI UN NUMERO? -17
01234567890123456789O
-17         -17

READY
```

```
RUN
DAMMI UN NUMERO? 99.34
01234567890123456789O
 99.34       99.34

READY
```

```
RUN
DAMMI UN NUMERO? -99.34
01234567890123456789O
-99.34      -99.34

READY
```

- ESEGUI IL PROGRAMMA 15 -

Comunque, ad ogni esecuzione la seconda cifra è trattata come stringa.

Nel programma successivo (Programma 16) teniamo conto del fatto che STR$(J) tratta J come stringa, perciò aggiungiamo il carattere J (come opposto al suo valore) alla fine di una stringa.

```
40 REM **ANCORA SU STR$**
50 REM **********
60 PRINT CHR$(147)
100 C$="":REM **VUOTA C$**
110 FOR J=1 TO 10
120 C$=C$+STR$(J)
125 REM **PER INCOLONNARE LE RIGHE**
130 IF J<10 THEN PRINT" ";
140 PRINT J;C$
150 NEXT J
160 END
```

Programma 16

```
RUN
 1     1
 2     1 2
 3     1 2 3
 4     1 2 3 4
 5     1 2 3 4 5
 6     1 2 3 4 5 6
 7     1 2 3 4 5 6 7
 8     1 2 3 4 5 6 7 8
 9     1 2 3 4 5 6 7 8 9
 10    1 2 3 4 5 6 7 8 9 10

READY
```

Quindi, ancora non abbiamo potuto legare i caratteri in posizioni adiacenti a causa dello spazio posto per il segno del numero.

- ESEGUI IL PROGRAMMA 16 -

Esercizio 4

Scrivi un programma per immettere una parola dalla tastiera, codificare ogni lettera come numero ed emettere il codice come sequenza di numeri. E cioè: la parola, che è una sequenza di lettere, và trasformata in una sequenza di numeri.

Ti diamo un suggerimento: definisci una lista di caratteri alfabetici che contenga l'intero alfabeto in ordine da A a Z. Ricorda l'istruzione DIM.

Prendi ogni lettera della parola e confrontala con gli elementi della lista di caratteri alfabetici.
Quando l'hai trovata, prendi l'indice e cioè il valore della posizione in cui si trova la lettera nella lista, trasformalo in stringa ed aggiungi tutti gli indici successivi alla stringa–codice.

Esercizio 5

Scrivi un programma per generare 20 parole a caso di 3 lettere (è interessante vedere quante volte fai girare questo programma per generare una parola che abbia un senso).

Compito 6

1. Scrivi un programma che distribuisce una mano di quante carte vuoi. Stampa la mano nella forma 1C, RP, 8Q, 6F..., dove P = picche, 1 = asso, C = cuori, Q = quadri, F = fiori, D = dieci, J = jack, e R = re. Ricorda che quando una carta viene distribuita una volta non può essere distribuita di nuovo.

   Ti diamo un suggerimento: scrivi il programma in più parti:
   - definisci il banco (quello che nell'esercizio 4 abbiamo chiamato lista)
   - distribuisci le carte (è più semplice usare il generatore RND 1–52;)
   - emetti l'output.

   Quando una carta viene distribuita, metti un segno (per es. un asterisco, *) in quella posizione per segnalare che non può essere usata di nuovo.

2. Scrivi un programma per simulare il gioco dell'oca usando un dado ed un percorso 4x4.

   Ti diamo un segguerimento: sebbene il percorso sia un quadrato, le caselle possono essere rappresentate in memoria dalla lista B(I):

   per es. B(1), B(2), B(3)...B(16)

   B(3) = +4 potrebbe voler dire al giocatore che si trova nella terza casella (B3) di andare avanti di 4 caselle.
   B(9) = –7 potrebbe voler dire al giocatore di indietreggiare di 7 caselle, dalla casella 9 (B9) etc.

Non dimenticare che il tuo ultimo lancio deve dare il numero giusto per completare il giro esattamente a 16.
Suggerimento: gioca qualche volta e stima il numero medio di lanci necessari per completare il percorso. Cambia la disposizione del percorso e fai qualche altra prova. Progetta un percorso più lungo, ecc.

Obiettivi dell'unità 6

Quando hai finito questa unità controlla se sai:

Usare RND per generare numeri casuali tra 0 e 1.	☐
Usare INT e RND per generare numeri casuali interi tra 0 e un dato intero N	☐
Simulare i lanci di una moneta	☐
Simulare i lanci di un dado	☐
Simulare il lancio di due dadi	☐
Usare liste di punteggi	☐
Abbreviare i programmi con ? e linee di istruzioni multiple	☐
Concatenare stringhe	☐
Usare STR$(X)	☐

Risposte alle DAV e agli esercizi

DAV1

```
10 PRINT CHR$(147)
20 FOR I=1 TO 6
30 LET N=6*RND(1)
40 PRINT N
50 NEXT I
60 END
```

Programma 17

DAV2

(a) 4; (b) 9; (c) –3; (d) –1; (e) 1

DAV3

```
RUN
-3.8         -4
-3.6         -4
-3.4         -4
-3.2         -4
-3           -3
-2.8         -3
-2.6         -3
-2.4         -3
-2.2         -3
-2           -2

READY
```

DAV4

```
RUN
 1.6          1
 1.8          1
 2            2
 2.2          2
 2.4          2
 2.6          2
 2.8          2
 3            3
 3.2          3

READY
```

Esercizio 1

Algoritmo descrittivo per il lancio di un dado

1. Inizio
2. Poni a zero le 6 locazioni totali del punteggio.
3. Inizia il ciclo dei 100 lanci
4. Genera un punteggio casuale dall'insieme (1, 2, 3, 4, 5, 6)
5. Incrementa il totale in relazione a questi punteggi

6. Se il contatore di ciclo < = 100 allora vai all'istruzione 7
7. Emetti il punteggio e il totale per ogni valore di punteggio
8. Stop.

```
10 REM **LANCIA UN DADO 100 VOLTE**
12 PRINT CHR$(147)
14 REM **SPAZIO PER CONTATORI**
15 DIM S(6)
17 PRINT"RISULTATI LANCIO DADO 100 VOLTE"
18 REM **AZZERA I CONTATORI**
20 FOR J=1 TO 6
30 LET S(J)=0
40 NEXT J
50 REM **CICLO DI GENERAZIONE**
60 FOR I=1 TO 100
70 LET S=INT(6*RND(1)+1)
80 LET S(S)=S(S)+1
85 REM **FINE DEL CICLO**
90 NEXT I
100 PRINT
110 PRINT"PUNTEGGIO","FREQUENZA"
115 REM **CICLO DI DISPLAY**
120 FOR K=1 TO 6
130 PRINT K,S(K)
140 NEXT K
150 END
```

Programma 18

```
RUN
RISULTATI LANCIO DADO 100 VOLTE

PUNTEGGIO FREQUENZA
 1         13
 2         20
 3         10
 4         20
 5         14
 6         23

READY
```

```
RUN
RISULTATI LANCIO DADO 100 VOLTE

PUNTEGGIO FREQUENZA
 1          17
 2          16
 3          17
 4          14
 5          19
 6          17

READY
```

```
RUN
RISULTATI LANCIO DADO 100 VOLTE

PUNTEGGIO FREQUENZA
 1          16
 2          11
 3          15
 4          25
 5          13
 6          20

READY
```

```
RUN
RISULTATI LANCIO DADO 100 VOLTE

PUNTEGGIO FREQUENZA
 1          12
 2          18
 3          26
 4          20
 5          10
 6          14

READY
```

- ESEGUI IL PROGRAMMA 18 -

Esercizio 2

```
10 REM **LANCIA 2 DADI 100 VOLTE**
12 PRINT CHR$(147)
14 REM **SPAZIO PER CONTATORI**
15 DIM S(15)
17 PRINT"RISULTATI LANCIO 2 DADI 100 VOLTE"
18 REM **AZZERA I CONTATORI**
20 FOR J=2 TO 12
30 LET S(J)=0
40 NEXT J
50 REM **CICLO DI GENERAZIONE**
60 FOR I=1 TO 100
70 LET S1=INT(6*RND(1)+1)
75 LET S2=INT(6*RND(1)+1)
80 LET S=S1+S2
82 LET S(S)=S(S)+1
85 REM **FINE DEL CICLO**
90 NEXT I
95 REM **OUTPUT DEI RISULTATI**
100 PRINT
110 PRINT"PUNTEGGIO","FREQUENZA"
115 REM **DISPLAY DEL CICLO**
120 FOR K=2 TO 12
130 PRINT K,S(K)
140 NEXT K
150 END
```

Programma 19

```
RUN
RISULTATI LANCIO 2 DADI 100 VOLTE

PUNTEGGIO FREQUENZA
 2         4
 3         3
 4         10
 5         11
 6         16
 7         17
 8         10
 9         13
```

```
 10           10
 11           2
 12           4

READY
```

- ESEGUI IL PROGRAMMA 19 -

E su 1000 lanci? Bene, cambia solo la linea 60 in 60 FOR I = 1 TO 1000 e la linea 17 PRINT ed otterrai:

```
RUN
RISULTATI LANCIO DADI 1000 VOLTE

PUNTEGGIO FREQUENZA
 2          23
 3          43
 4          105
 5          116
 6          148
 7          146
 8          140
 9          110
 10         85
 11         50
 12         34

READY
```

Suggerimenti per programmi futuri

1. Come affronterai la stampa di un diagramma di frequenza su dati, che vadano oltre la fine della riga, sullo schermo?
2. Abbiamo bisogno di una routine generale che scali i caratteri sullo schermo e che li aggiusti a differenti lunghezze di riga, permettendo così un uso migliore della lunghezza totale della pagina o dello schermo.

Esercizio 3

```
10 REM **DIAGRAMMA DI FREQUENZA**
12 PRINT CHR$(147)
14 REM **SPAZIO PER CONTATORI**
15 DIM S(15)
17 PRINT"RISULTATI LANCIO 2 DADI 100 VOLTE"
18 REM **AZZERA I CONTATORI**
20 FOR J=2 TO 12
30 LET S(J)=0
40 NEXT J
50 REM **CICLO DI GENERAZIONE**
60 FOR I=1 TO 100
70 LET S1=INT(6*RND(1)+1)
75 LET S2=INT(6*RND(1)+1)
80 LET S=S1+S2
82 LET S(S)=S(S)+1
85 REM **FINE CICLO**
90 NEXT I
95 REM **OUTPUT DEI RISULTATI**
100 PRINT
110 PRINT"DIAGRAMMA DI FREQUENZA"
115 PRINT
120 FOR K=2 TO 12
130 PRINT K;TAB(5);S(K);TAB(10);
140 IF S(K)=0 THEN 180
150 FOR L=1 TO S(K)
160 PRINT"*";
170 NEXT L
180 REM **SALTA QUI SE FREQ. E' ZERO**
190 PRINT
200 NEXT K
210 END
```

Programma 20

```
RUN
RISULTATI LANCIO 2 DADI 100 VOLTE

DIAGRAMMA DI FREQUENZA

 2    2    **
 3    7    *******
 4    8    ********
 5    9    *********
 6    16   ****************
 7    13   *************
 8    15   ***************
 9    16   ****************
 10   7    *******
 11   3    ***
 12   4    ****

READY
```

- ESEGUI IL PROGRAMMA 20 -

DAV5

La soluzione che segue è solo uno dei molti modi possibili per abbreviare il programma.

```
10 REM **DIAGRAMMA DI FREQUENZA**
15 DIM S(15):PRINT CHR$(147)
17 PRINT"RISULTATI LANCIO 2 DADI 100 VOLTE"
20 FOR J=2 TO 12:S(J)=0:NEXTJ
60 FOR I=1 TO 100
70 LET S1=INT(6*RND(1)+1):S2=INT(6*RND(1)+1)
80 LET S=S1+S2:S(S)=S(S)+1
90 NEXT I
100 PRINT:PRINT"DIAGRAMMA DI FREQUENZA"
120 FOR K=2 TO 12
130 PRINT K;TAB(5);S(K);TAB(10);
140 IF S(K)=0 THEN 180
```

```
150 FOR L=1 TO S(K):PRINT"*";:NEXT L
190 PRINT
200 NEXT K
210 END
```

Programma 21

```
RUN
RISULTATI LANCIO 2 DADI 100 VOLTE

DIAGRAMMA DI FREQUENZA
 2     2    **
 3     5    *****
 4     4    ****
 5     14   **************
 6     19   *******************
 7     19   *******************
 8     15   ***************
 9     7    *******
 10    7    *******
 11    4    ****
 12    4    ****

READY
```

DAV6

```
10 REM **PLURALI IN INGLESE**
20 LET A$="S"
30 INPUT B$
40 PRINT B$+A$
50 END
```

Programma 22

Esercizio 4

```
10 REM **SEMPLICE CODIFICA**
20 PRINT CHR$(147)
```

```
30 DIM A$(26)
40 FOR I=1 TO 26
50 READ A$(I)
60 NEXT I
70 DATA A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P
80 DATA Q,R,S,T,U,V,W,X,Y,Z
100 INPUT"ALTRA PAROLA DA CODIFICARE";W$
110 L=LEN(W$)
120 FOR J=1 TO L
130 I=1
140 IF MID$(W$,J,1)=A$(I) THEN 200
150 I=I+1
160 GOTO 140
200 C$=C$+STR$(I)+""
210 NEXT J
220 PRINT:PRINT C$
230 END
```

Lines 30–80: definisce la directory

Lines 130–150: confronta ogni lettera della parola con ogni lettera nella lista finché non la trova

Line 200: poi aggiunge l'indice in forma di stringa alla stringa indice

Programma 23

```
RUN
ALTRA PAROLA DA CODIFICARE? COMPUTER

 3 15 13 16 21 20 5 18

READY


RUN
ALTRA PAROLA DA CODIFICARE? PARLAMENTO

 16 1 18 12 1 13 5 14 20 15

READY


RUN
ALTRA PAROLA DA CODIFICARE? PROFESSIONE

 16 18 15 6 5 19 19 9 15 14 5

READY
```

- ESEGUI IL PROGRAMMA 23 -

Esercizio 5

```
10 REM **PAROLE CASUALI DE TRE LETTERE**
20 DIM A$(26)
30 FOR I=1 TO 26:REM **LISTA IN ORDINE ALFABETICO**
40 READ A$(I):NEXT I
50 DATA A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,
60 DATA Q,R,S,T,U,V,W,X,Y,Z
70 PRINT CHR$(147)
80 INPUT"UN'ALTRA LISTA";R$
90 IF R$<>"YES" THEN 190
100 FOR K=1 TO 20:REM **20 PAROLE**
110 W$=" ":REM **VUOTA LE PAROLE-STRINGA ALL'INIZIO**
120 FOR J=1 TO 3:REM **INIZIO DELLA PAROLA**
130 X=INT(26*RND(I)+1)
140 W$=W$+A$(X)
150 NEXT J:REM **FINE DELLA PAROLA**
160 PRINT W$:REM **STAMPA DELLA PAROLA**
170 NEXT K:REM **TORNA ALLA PAROLA SUCCESSIVA**
180 GOTO 80
190 END
```

Programma 24

```
RUN
UN'ALTRA LISTA? SI
 TPM
 PUD
 LBQ
 PFV
 JOV
 FXI
 TTL
 AAW
 JB
 DXX
 PWH
 IMQ
 WJR
 BIW
 WU
 OMJ
 XMF
```

```
 UMM
 JTA
 IOS
UN'ALTRA LISTA?
```

Quante volte devi far girare il programma per ottenere una parola vera?

- ESEGUI IL PROGRAMMA 24 -

UNITÀ 7(1)
Grafici, suoni e colori

(1) NOTE PER LA LETTURA DEI CARATTERI GRAFICI

Queste note sono da applicarsi a tutti quei caratteri che sono stati racchiusi tra parentesi graffe.
Queste parentesi sono state scelte come delimitatori perché non sono disponibili sulla tastiera del Commodore 64.
Se i primi due caratteri racchiusi tra graffe sono:

CO (o CBM)
CT
SH

si dovrà intendere che il carattere immediatamente seguente a questi dovrà essere digitato insieme al tasto:

CO = COMMODORE,
CT = CONTROL,
SH = SHIFT.

7.1 Introduzione

Fin qui abbiamo usato il Commodore 64 soltanto per manipolare numeri e testi, nei colori standard dello schermo, blue chiaro su blue scuro.
Comunque, è più divertente un microcomputer con una vasta gamma di colori sullo schermo, grafici e suoni.
Ovviamente i giochi di tipo tradizionale sono più divertenti a colori, specie se accompagnati da suoni e rumori adatti, generati quando gli "invasori" o le "navi" vengono distrutti. Ma anche la grafica e i programmi educativi, per esempio, sono più chiari e più facili da usare con colori e suoni appropriati.

7.2 Colore: il margine

Ci sono tre aree principali dello schermo che possono avere un colore. La prima è il margine dello schermo. Quando accendi il computer, puoi vedere quest'area colorata in azzurro chiaro.
Il Commodore 64 usa l'istruzione POKE del BASIC per immettere codici di controllo del colore direttamente sul computer. L'istruzione ha la forma:

POKE locazione di memoria, codice di controllo

La locazione di memoria 53280 controlla il colore del margine. Se esegui il programma seguente potrai vedere l'effetto dell'istruzione POKE in questa area.

```
10 REM ****************
20 REM **COLORI DEL MARGINE**
30 REM ****************
40 PRINT CHR$(147)
50 FOR X=0 TO 15
60 POKE 53280,X
70 FOR Z=1 TO 500
80 NEXT Z
90 REM **PAUSA PER RALLENTARE IL PROGRAMMA**
100 REM  TANTO DA VEDER APPARIRE
110 REM  OGNI COLORE**
120 NEXT X
130 END
```

Programma 1 - Colori del margine.

- ESEGUI IL PROGRAMMA 1 -

Il ciclo FOR...NEXT... nelle linee 70 e 80 rallenta la velocità con cui l'output appare sullo schermo in modo tale da permetterti di vedere cambiare i colori del margine.
Senza queste due linee, la velocità di cambiamento dei margini sarebbe troppo rapida per essere vista chiaramente. Sebbene il ciclo più interno FOR...NEXT... non contenga istruzioni di esecuzione, richiede al computer un po' di tempo per contare fino al numero limite, la qual cosa rallenta il computer stesso.
Avrai visto l'area del margine dello schermo cambiare ciclicamente in 16 colori, nell'ordine: nero, bianco, rosso, azzurro, porpora, verde, blu, giallo, arancio, marrone, rosso chiaro, grigio 1, grigio 2, verde chiaro e grigio 3. Questi colori corrispondono ai numeri da 0 a 15 incluso.
Come esercizio, perché non usare

POKE 53280, numero di colore

per aggiungere un margine colorato ad alcuni dei programmi che hai già battuto durante questo corso?

0 nero	8 arancio
1 bianco	9 marrone
2 rosso	10 rosso chiaro
3 azzurro	11 grigio 1
4 porpora	12 grigio 2
5 verde	13 verde chiaro
6 blu	14 blu chiaro
7 giallo	15 grigio 3

Per tornare dallo schermo colorato a quello normale azzurro/blu, premi RUN STOP e RESTORE

7.3 Colore: la carta o lo sfondo

La parte principale dello schermo, all'interno del margine, è detta sfondo, o "carta". Questa è l'area su cui hai visto apparire i tuoi programmi e i risultati. Il colore è controllato dall'istruzione POKE, che mette il numero del colore richiesto alla locazione di memoria 53281.
Questo programma stampa lo schermo in ognuno dei 16 colori disponibili, a turno.

```
10 REM *****************
20 REM **COLORI DELLO SFONDO**
30 REM *****************
40 PRINT CHR$(147)
50 FOR X=0 TO 15
60 POKE 53281,X
70 FOR Z=1 TO 500
80 NEXT Z
90 NEXT X
100 END
```

Programma 2 - Colori dello sfondo

- ESEGUI IL PROGRAMMA 2 -

Non è necessario pulire lo schermo del Commodore 64 prima di cambiare i colori dello sfondo.

7.4 Colore: L'inchiostro

Proprio come il colore dello sfondo può essere chiamato "carta", il colore dei caratteri stampati sullo schermo può essere chiamato "inchiostro".

Fino a questa unità didattica abbiamo usato un inchiostro azzurro o blu scuro: questi sono i valori di "default", e cioè i colori forniti dal computer quando viene acceso o quando un programma esistente viene richiamato dalla memoria con la funzione NEW o RUN STOP/RESTORE.
Per cominciare, osserviamo la produzione di colori con l'istruzione.

PRINT CHR$ (numero di codice)

Come abbiamo visto alla fine dell'unità 3, questa istruzione BASIC può essere usata per stampare un qualsiasi carattere ASCII disponibile sul Commodore 64. L'appendice F del manuale dell'utente illustra i codici dei caratteri.

CODICE ASCII	COLORE
5	Bianco
28	Rosso
30	Grigio
31	Blu
144	Nero
156	Porpora
158	Giallo
159	Azzurro–Verde

Come vedi, i codici di colore sono sparsi in luoghi diversi.
Solo questi otto colori sono indicati sulla tastiera del Commodore.
Questo programma stampa strisce colorate sullo schermo:

```
10 REM ****************
20 REM **STRISCE COLORATE**
30 REM ****************
40 PRINT CHR$(147);
50 FOR X=0 TO 7
60 READ CL
65 PRINT CHR$(CL);
70 PRINT CHR$(18);
80 FOR Y=1 TO 40
100 PRINT CHR$(32);
110 NEXT Y
120 NEXT X
130 END
140 DATA 5,28,30,31,156,158,159,144
```

Programma 3 – Strisce colorate.

ESEGUI IL PROGRAMMA 3 -

La linea 70 usa il codice di carattere ASCII 18 (RVS ON) – che stampa i 40 spazi che compongono una linea (codice di carattere ASCII 160) in negativo, per cui l'effetto è quello di stampare una striscia del colore richiesto. Possiamo stampare anche caratteri colorati su strisce colorate.

```
10 REM ***************
20 REM **MESSAGGI COLORATI**
30 REM ***************
```

```
40 PRINT CHR$(147)
50 FOR Y=1 TO 3
60 FOR X=0 TO 7
70 READ CL
80 PRINT CHR$(CL)
90 PRINT"BUONGIORNO A TE PROGRAMMATORE"
100 NEXT X
110 RESTORE
120 NEXT Y
130 GOTO 130
140 DATA 5,28,30,31,156,158,159,144
```

Programma 4 - Messaggi colorati

- ESEGUI IL PROGRAMMA 4 -

Dall'esecuzione di questo programma vedrai che alcune combinazioni di inchiostro e carta sono di più facile lettura rispetto ad altre. O peggio, senza un monitor a colori può verificarsi una perdita di qualità dell'immagine nel caso di alcune combinazioni di colore che rendono difficile la lettura dei caratteri. E' importante quindi una scelta saggia e oculata delle combinazioni di colori. Presto scoprirai, nelle tue rappresentazioni, quali combinazioni di colori meglio si adattano, per tentativi ed errori.
Se speri di produrre programmi che siano usati da altre persone, ti sia ben chiaro che non tutti useranno uno schermo colorato. Combinazioni di colore di inchiostro e carta devono avere un buon contrasto anche in un monitor in bianco e nero. Per esempio blu e rosso, in blocchi compatti, sono chiaramente distinguibili, ma fanno poco contrasto con il bianco e il nero, essendo vicini l'uno all'altro nella scala dei contrasti.

Esempio 1

Usa il metodo del programma 3 per produrre un arcobaleno a 6 colori. Il nostro arcobaleno sarà più bello su un margine scuro, rendiamo quindi il margine e il fondo nero (colore 0).

```
50 POKE 53281,0
60 POKE 53280,0
```

Altrimenti, può essere adottato lo stesso metodo del programma 3. Abbiamo 6 fasce di colore. Inseriamo il ciclo di controllo FOR...NEXT...

```
70 FOR X=1 TO 6
160 NEXT X
```

Lo schermo contiene 25 linee, così lo possiamo dividere in 6 fasce di 4 linee (totale 24 linee). Stampa ogni striscia colorata 4 volte, usando un ciclo nidificato nel ciclo FOR...NEXT...:

```
100 FOR Z=1 TO 4
150 NEXT Z
```

L'ordine dei colori dell'arcobaleno è:
rosso, giallo, verde, celeste, indaco, viola.
Noi non li abbiamo tutti e una soluzione possibile potrebbe essere:
rosso, giallo, verde, azzurro, blu, porpora.
Così la nostra istruzione DATA sarà
180 DATA 28, 158, 30, 159, 31, 156

Possiamo ora inserirla nel resto del programma:

```
10 REM *****************
20 REM **ARCOBALENO DI 6 COLORI**
30 REM *****************
40 PRINT CHR$(147)
50 POKE 53281,0
60 POKE 53280,0
70 FOR X=1 TO 6
80 READ CL
90 PRINT CHR$(CL);
100 FOR Z=1 TO 4
110 PRINT CHR$(18);
120 FOR Y=1 TO 40
130 PRINT CHR$(160);
140 NEXT Y
150 NEXT Z
160 NEXT X
170 GOTO 170
180 DATA 28,158,30,159,31,156
```

Programma 5 – Arcobaleno di 6 colori.

- ESEGUI IL PROGRAMMA 5 -

La linea 170 deve prevenire l'apparizione sullo schermo del messaggio "READY" per non rovinare l'effetto. Batti RUN STOP per fermare il programma.

7.5 Il colore con il comando POKE

Lo schermo del Commodore 64 è controllato da un blocco di 1000 locazioni di memoria, da 1024 a 2023. Lo schermo ha un'ampiezza di 40 caratteri e una profondità di 25 linee. (40 X 25 = 1000).
Le locazioni sono disposte in una griglia (v. Appendice G del Manuale per l'utente).
Il colore delle 1000 locazioni possibili di carattere sullo schermo è controllato da un altro blocco di memoria, come illustrato nell'Appendice G del Manuale per l'utente.

Prova a migliorare il programma arcobaleno.

```
10 REM *****************
20 REM **ARCOBALENO DI 7 COLORI**
30 REM *****************
40 PRINT CHR$(147)
50 POKE 53281,0
60 POKE 53280,0
70 FOR X=1024 TO 2023
80 POKE X,160
90 NEXT X
100 FOR C=0 TO 6
110 READ CL
120 FOR X=55296+120*C TO 55415+120*C
130 POKE X,CL
140 NEXT X
150 NEXT C
160 GOTO 160
170 DATA 2,8,7,5,14,6,4,
```

Programma 6 – Arcobaleno a 7 colori

- ESEGUI IL PROGRAMMA 6 -

Come prima, le linee 50 e 60 rendono neri il fondo e il margine. Le linee da 70 a 90 mettono il carattere "spazio" (ASCII 160) su tutte le locazioni dello schermo.

Il ciclo di elaborazione principale si trova nelle linee da 100 a 150. La linea 120 si riferisce alla mappa di memoria di colore. Questo fa sì che vengano colorati gruppi di 3 linee adiacenti sullo schermo (3x7=21, che entra nelle 25 linee dello schermo).
Il DATA (linea 170) usa i codici di colore che sono: 2 (rosso), 8 (arancio), 7 (giallo), 5 (verde), 14 (azzurro), 6 (blu), 4 (porpora).
Se la linea 120 ti sembra strana, prova a bloccare l'esecuzione del programma. Quando C è 0, la prima striscia va da 55296 a 55415, quando C è 1, la seconda va da 55416 a 55535 e così via.

7.6 Luci colorate

Ora che sappiamo come il Commodore 64 lavora con i colori, vediamo come un televisore a colori o un monitor producono i colori che vogliamo. Fai eseguire (RUN) il programma 7, luci colorate, e avrai una dimostrazione della spiegazionee che segue:

```
10 REM ***************
20 REM **LUCI COLORATE**
30 REM ***************
40 POKE 53280,0
50 POKE 53281,0
60 PRINT CHR$(147)
70 REM ***************
80 FOR X=1024 TO 2023
90 POKE X,160
100 NEXT X
110 REM ***************
120 PRINT"COCKTAIL DI COLORI"
130 REM **POSIZIONARE I PUNTINI LUMINOSI**
140 FOR X=0 TO 15
150 FOR Y=55508+40*X TO 55523+40*X
160 READ C
170 POKE Y,C
180 NEXT Y
190 NEXT X
200 GOTO 200
210 DATA 2,2,2,2,2,2,2,2,2,2,0,0,0,0,0,0,
220 DATA 2,2,2,2,2,2,2,2,2,2,0,0,0,0,0,0,
230 DATA 2,2,2,2,2,2,7,7,7,7,5,5,5,5,5,5,
```

```
240 DATA 2,2,2,2,2,2,7,7,7,7,5,5,5,5,5,5,
250 DATA 2,2,2,2,2,2,7,7,7,7,5,5,5,5,5,5,
260 DATA 2,2,2,2,2,2,7,7,7,7,5,5,5,5,5,5,
290 DATA 2,2,4,4,4,4,1,1,1,1,3,3,5,5,5,5,
300 DATA 2,2,4,4,4,4,1,1,1,1,3,3,5,5,5,5,
310 DATA 2,2,4,4,4,4,1,1,1,1,3,3,5,5,5,5,
320 DATA 2,2,4,4,4,4,1,1,1,1,3,3,5,5,5,5,
330 DATA 0,0,6,6,6,6,3,3,3,3,3,3,5,5,5,5,
340 DATA 0,0,6,6,6,6,3,3,3,3,3,3,5,5,5,5,
350 DATA 0,0,6,6,6,6,6,6,6,6,6,6,0,0,0,0,
360 DATA 0,0,6,6,6,6,6,6,6,6,6,6,0,0,0,0,
370 DATA 0,0,6,6,6,6,6,6,6,6,6,6,0,0,0,0,
380 DATA 0,0,6,6,6,6,6,6,6,6,6,6,0,0,0,0,
```

Programma 7 – Luci colorate

- ESEGUI IL PROGRAMMA 7 -

Le linee di dati da 210 a 380 appaiono qui come un disegno dipinto da numeri. Se hai colorato ogni numero con il suo colore corrispondente, avrai lo stesso disegno che appare sullo schermo. Questi mostra il potere dell'istruzione POKE direttamente all'interno della memoria dello schermo.
Vedrai che il programma produce tre luci intermittenti rettangolari che si sovrappongono in parte: quella a sinistra è rossa, quella a destra è verde e quella in basso è blu. Quando le luci rossa e verde si incontrano, producono il giallo, il blu e il verde producono l'azzurro e il rosso e il blu producono il porpora. Quando tutte e tre le luci si incontrano, producono una luce bianca. Quando non c'è luce, otteniamo il nero.
Lo schermo del televisore a colori, è sensibile solo al rosso, verde e blu e produce le altre tonalità, per sovrapposizione (o se preferisci dalla fusione) di piccoli punti di questi tre colori primari.

DAV1

Quali colori saranno generati dalle istruzioni che seguono e in quali aree dello schermo?

(a) POKE 53280,7 (b) POKE 53281,6
(c) PRINT CHR$(156)

Esempio 2

Aggiungi un colore al seguente programma di controllo di tabelle, per renderlo più efficace:

```
10 REM **CONTROLLO DI TABELLE**
20 LET C=0
30 LET D=0
40 PRINT"{SHCLRHOME}{BASSO}{BASSO}{BASSO}{BASSO}
{BASSO}{BASSO}{BASSO}{DESTRA}{DESTRA}{DESTRA}
{DESTRA}{DESTRA}{DESTRA}{DESTRA}{DESTRA}CONTR
OLLA LE TUE TABELLE"
50 PRINT"{BASSO}{BASSO}{BASSO}{BASSO}{DESTRA}
{DESTRA}{DESTRA}{DESTRA}BATTI LA TUA RISPOSTA
, POI PREMI"
60 PRINT"{DESTRA}{DESTRA}{DESTRA}{DESTRA}IL T
ASTO RETURN."
70 PRINT"{BASSO}{BASSO}{BASSO}{BASSO}{DESTRA}
{DESTRA}{DESTRA}{DESTRA}{DESTRA}{DESTRA}{DESTRA}
{DESTRA}{DESTRA}{DESTRA}{DESTRA}{DESTRA}{DESTRA}
{DESTRA}PREMI UN TASTO"
80 PRINT"{DESTRA}{DESTRA}{DESTRA}{DESTRA}{DESTRA}
{DESTRA}{DESTRA}QUANDO SEI PRONTO A PARTIRE"
90 GET A$
100 IF A$>"A" AND A$<="Z" THEN 130
110 GOTO 90
130 LET A=INT(RND(1)*13)
140 LET B=INT(RND(1)*13)
150 LET E=0
160 LET D=D+1
170 PRINT"{SHCLRHOME}{BASSO}{BASSO}{BASSO}{BASSO}
{BASSO}{BASSO}{BASSO}{DESTRA}{DESTRA}{DESTRA}
{DESTRA}{DESTRA}{DESTRA}{DESTRA}{DESTRA}{DESTRA}
{DESTRA}{DESTRA}{DESTRA}";A;"X";B;"=?"
180 PRINT"{BASSO}{BASSO}{BASSO}{BASSO}{BASSO}
{BASSO}{BASSO}{DESTRA}BATTI LA TUA RISPOSTA-P
OI PREMI RETURN"
185 PRINT"{BASSO}{DESTRA}{DESTRA}{DESTRA}{DESTRA}
{DESTRA}{DESTRA}{DESTRA}{DESTRA}{DESTRA}{DESTRA}
{DESTRA}{DESTRA}{DESTRA}";
190 INPUT F
```

```
200 PRINT"{SHCLRHOME}{BASSO}{BASSO}{BASSO}{BASSO}
{BASSO}{BASSO}{BASSO}{DESTRA}{DESTRA}{DESTRA}
{DESTRA}{DESTRA}{DESTRA}{DESTRA}{DESTRA}{DESTRA}
{DESTRA}{DESTRA}{DESTRA}";A"X";B;"=";F
210 IF F<>A*B THEN GOTO 330
220 IF E>0 THEN GOTO 240
230 LET C=C+1
240 PRINT"{BASSO}{BASSO}{BASSO}{BASSO}{BASSO}
{BASSO}{BASSO}{DESTRA}GIUSTO!! IL TUO PUNTEGG
IO ORA E'";C;"/";D
250 FOR Z=1 TO 2000
260 NEXT Z
270 IF D<10 THEN GOTO 130
280 PRINT"{SHCLRHOME}{BASSO}{BASSO}{BASSO}{BASSO}
{BASSO}{DESTRA}{DESTRA}{DESTRA}{DESTRA}{DESTRA}
{DESTRA}{DESTRA}IL PUNTEGGIO E' ";C;" SU 10"
290 PRINT"{BASSO}{BASSO}{BASSO}{BASSO}{DESTRA}
{DESTRA}{DESTRA}{DESTRA}{DESTRA}{DESTRA}{DESTRA}
{DESTRA}{DESTRA}{DESTRA}{DESTRA}{DESTRA}PER U
N ALTRO GIRO:"
300 PRINT"{BASSO}{BASSO}{DESTRA}{DESTRA}{DESTRA}
{DESTRA}{DESTRA}{DESTRA}{DESTRA}{DESTRA}{DESTRA}
{DESTRA}{DESTRA}{DESTRA}{DESTRA}{DESTRA}{DESTRA}
BATTI RUN"
310 PRINT"{BASSO}{BASSO}{DESTRA}{DESTRA}{DESTRA}
{DESTRA}{DESTRA}{DESTRA}{DESTRA}{DESTRA}POI P
REMI IL TASTO RETURN"
320 PRINT"{BASSO}{BASSO}{BASSO}{BASSO}"
325 END
330 LET E=E+1
340 IF E>=3 THEN GOTO 390
350 PRINT"{BASSO}{BASSO}{BASSO}{BASSO}{BASSO}
{BASSO}{BASSO}{DESTRA}{DESTRA}{DESTRA}{DESTRA}
{DESTRA}{DESTRA}{DESTRA}{DESTRA}{DESTRA}{DESTRA}
SBAGLIATO-RIPROVA"
360 FOR Z=1 TO 2000
370 NEXT Z
380 GOTO 170
390 FOR Z=1 TO 2000
400 NEXT Z
410 PRINT"{SHCLRHOME}{BASSO}{BASSO}{BASSO}{BASSO}
{BASSO}{BASSO}{BASSO}{DESTRA}{DESTRA}{DESTRA}
```

```
{DESTRA}{DESTRA}{DESTRA}{DESTRA}{DESTRA}{DESTRA}
{DESTRA}{DESTRA}{DESTRA}{DESTRA}";A;"X";B;"="
;A*B
420 PRINT"{BASSO}{BASSO}{BASSO}{BASSO}{BASSO}
{BASSO}{BASSO}{DESTRA}{DESTRA}{DESTRA}{DESTRA}
{DESTRA}QUESTA E' LA RISPOSTA CORRETTA"
440 FOR Z=1 TO 5000
450 NEXT Z
460 LET E=0
470 GOTO 270

READY.
```

Programma 8 – Controllo di tabelle

- ESEGUI IL PROGRAMMA 8 -

```
RUN

          CONTROLLA LE TUE RISPOSTE

      BATTI LA TUA RISPOSTA, POI PREMI
             IL TASTO RETURN.
              PREMI UN TASTO
       QUANDO SEI PRONTO A PARTIRE

               12 X 10 =?
  BATTI LA TUA RISPOSTA-POI PREMI RETURN
                 ? 120

               12 X 10 = 120
  GIUSTO!! IL TUO PUNTEGGIO ORA E' 1 / 1

               12 X 10 = 3
            SBAGLIATO-RIPROVA

               12 X 10 = 120
       QUESTA E' LA RISPOSTA CORRETTA


          IL PUNTEGGIO E' 1 SU 10
              PER UN ALTRO GIRO:
                 BATTI RUN
          POI PREMI IL TASTO RETURN
```

Questo programma di test sulla tabellina della moltiplicazione dà un insieme di domande generate casualmente nello spazio da 0 x 0 a 12 x 12.
Se viene battuta la risposta errata, viene data un'altra possibilità.
Dopo tre risposte errate, appare automaticamente la risposta corretta.
Questo è un programma dallo stile abbastanza "amichevole", ma potrebbe essere migliorato con l'uso del colore.
Per prima cosa, aggiungi un margine colorato allo schermo.

```
15 POKE 53280,5
```

che darà uno schermo verde. Le linee seguenti:

```
311 FOR Y=1 TO 20
312 POKE 53280,2
313 FOR Z=1 TO 100
314 NEXT Z
315 POKE 53280,6
316 FOR Z=1 TO 100
317 NEXT Z
318 NEXT Y
```

cambiano ciclamente il colore del margine in blu e rosso per venti volte, per mettere in evidenza il risultato del punteggio. Ora fai eseguire (RUN) il programma e dicci se ti pare che questo piccolo cambiamento di colore migliori il programma stesso o no. Prova da solo per vedere quali colori possono essere più adatti.
Torneremo su questo programma più avanti, nel corso di questa unità, quando parleremo di suoni.
I tuoi programmi saranno più attraenti se il colore è applicato alla grafica – cioè disegni sullo schermo – e specialmente al movimento.

7.7 Grafica e movimento

Vediamo un esempio di come produrre un semplice movimento grafico.
L'immagine o i simboli grafici, sul Commodore 64 si trovano sulla tastiera e vi si accede attraverso i tasti SHIFT, CTRL e il tasto "Logo Commodore", battuti insieme con altri tasti.

Esempio 3

Programma di movimento dell'autobus.
Questo programma muoverà sullo schermo un autobus piuttosto grosso:

```
10 REM ***************
20 REM **MOVIMENTO DEL BUS**
30 REM ***************
40 LET B$(1)="{RVS ON}{CBM D}{5 CBM
    F}{RVS OFF}"
50 LET B$(2)="{RVS ON}{7 SPC}{RVS O
   FF}"
60 LET B$(3)=" O{3 SPC}O"
70 LET C$="{39 SPC}"
80 FOR M=1 TO 32
90 PRINT"{CLR}{5 CUR.GIU}"
120 FOR K=1 TO 3
130 PRINT LEFT$(C$,M);B$(K)
140 NEXT K
150 FOR Z=1 TO 25
160 NEXT Z
170 NEXT M
180 END
```

Programma 9 – Movimento dell'autobus

- ESEGUI IL PROGRAMMA 9 -

Per aiutarti a riprodurre la figura dell'autobus, i simboli grafici all'interno delle virgolette vengono dati da:

Linea 40: CTRL/9 (RVS ON)
Commodore/D
Commodore/F (ripeti per altre 4 volte)
CTRL/0 (RVS OFF)

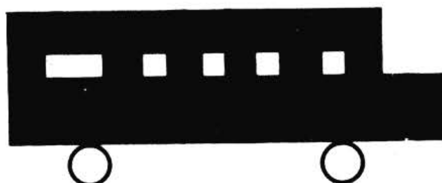
Linea 50: CTRL/9(RVS ON)
Barra di spaziatura o, come dicono in inglese i bravi programmatori "space bar" (ripeti per altre 6 volte)
CTRL/0 (RVS OFF)

Linea 60: space bar
o (lettera o, non zero)
space bar (ripeti per altre 2 volte)
O

Linea 70: 39 spazi fra le due virgolette

Linea 90: SHIFT/CLR HOME (pulire lo schermo)
cursore verticale (ripeti per 4 volte)
(per muovere il cursore di 5 righe in basso)
Quando batterai RUN, sullo schermo apparirà:

L'autobus sarà blu chiaro/blu scuro, e si muoverà attraverso lo schermo.



Magia!
Le linee 80 e 170 disegnano il bus, puliscono lo schermo, disegnano il bus un quadratino a destra, puliscono lo schermo, e così via. Le linee 120 e 140 stampano gli spazi davanti al bus e ridisegnano il bus come se si muovesse sullo schermo. Questo è esattamente il modo in cui avviene il movimento dei cartoni animati: la leggera differenza nella posizione di ogni disegno, accompagnata da un rapido movimento, da una figura alla successiva, ci fa pensare che l'immagine si stia effettivamente muovendo.
Adattiamo la velocità del bus attraverso il ciclo delle linee 150 e 160; il numero dopo TO alla linea 150 fa muovere più o meno velocemente il bus. Cerca di adattare la linea 150 fino a raggiungere la velocità desiderata.
Molte compagnie di trasporti non hanno autobus blu, perciò facciamolo diventare rosso. Ma le gomme delle ruote sono nere. Aggiungiamo i codici di colore all'interno delle stringhe, alle linee 40, 50 e 60.

Metti il primo elemento nella linea 40 fra virgolette

CTRL/3 (che significa rosso)

e fai lo stesso con la linea 50. Comincia la linea 60 con CTRL/1 (che significa nero)

Aggiungi

```
180 PRINT "{CT7}"
```

per ritornare dalla stampa colorata alla normale, alla fine del programma. Bene, questo è tutto molto carino, ma cosa succede alla strada su cui cammina il bus? Nessun problema basta, aggiungere

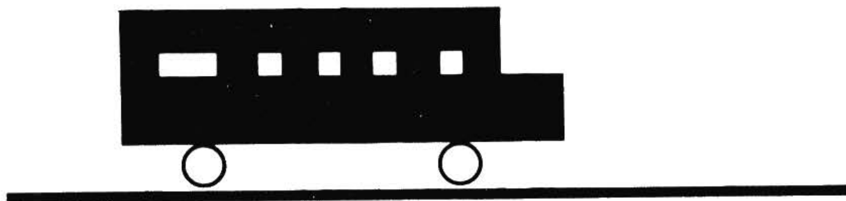
Linea 100 CTRL/1 + Commodore/U (ripetuto 38 volte)

Linea 110 SHIFT/cursore verticale (ripetuto 5 volte)

```
10 REM ****************
20 REM **MOVIMENTO DEL BUS**
30 REM ****************
40 LET B$(1)="{RED}{RVS ON}{CBM D}{
   5 CBM F}{RVS OFF}"
50 LET B$(2)="{RED}{RVS ON}{7 SPC}{
   RVS OFF}"
60 LET B$(3)="{BLACK} O{3 SPC}O"
70 LET C$="{38 SPC}"
80 FOR M=1 TO 32
90 PRINT"{CLR}{5 CUR.GIU}"
100 PRINT"{BLACK}";
105 PRINT"{38 CBM U}"
110 PRINT"{5 CUR.SU}"
120 FOR K=1 TO 3
130 PRINT LEFT$(C$,M);B$(K)
140 NEXT K
150 FOR Z=1 TO 25
160 NEXT Z
170 NEXT M
180 PRINT"{SH Z}"
```

Programma 10 – Un autobus si muove sulla strada

- ESEGUI IL PROGRAMMA 10 -



Disegno tipico dell'autobus in movimento. L'autobus è disegnato in rosso.
I principi che abbiamo evidenziato in questo esempio ti permettevano di disegnare e colorare le tue figure. Ora sperimenta con i simboli della grafica, per far muovere la figura. Non dimenticare di estendere la linea vuota (linea 130) al lato sinistro di ogni linea della figura, così che la figura cancelli la sua coda nei movimenti.
Più avanti, in questa unità, ritorneremo al nostro autobus, per aggiungere il suono.

Esempio 4

Disegno con macchie di colore

La funzione GET permette all'utente del computer di inserire il singolo tasto senza aver premuto RETURN dopo che il tasto è stato battuto. Quando viene usato GET, generalmente è necessario controllare se è stato battuto un tasto e rispondere immediatamente. La risposta può essere una di queste tre:

- esegue l'appropriata azione per quel tasto, come stabilito dal programma.
- dà un messaggio di errore "questa volta la battuta non è valida. Riprova".
- oppure per tutti i tasti nulli, aspetta che venga pigiato un tasto valido.

Nel programma 11, soltanto i tasti 5, 6, 7, 8, e 9 sono validi. Se qualsiasi altro tasto è premuto, non deve essere eseguita nessuna azione.
Tutto ciò che succede è che la linea 190 ridirige il programma alla linea 55 per aspettare un tasto valido.

Questo programma usa il tasto 5 per muovere il cursore a sinistra, 6 per muoverlo in basso e 7 per muoverlo in alto, 8 per muoverlo a destra e 9 per cambiare colore. Niente ti impedisce di usare altri tasti per queste funzioni. Le linee 100, 120, 140, e 160 riguardano i casi in cui la macchia di colore nelle linee 55 e 60 raggiunge l'orlo dello schermo. Per ogni linea il colore cambia se batti il tasto 9. Se arrivi al colore 15 (grigio 3), allora il colore ritorna indietro a 0 (nero) con la linea 180.

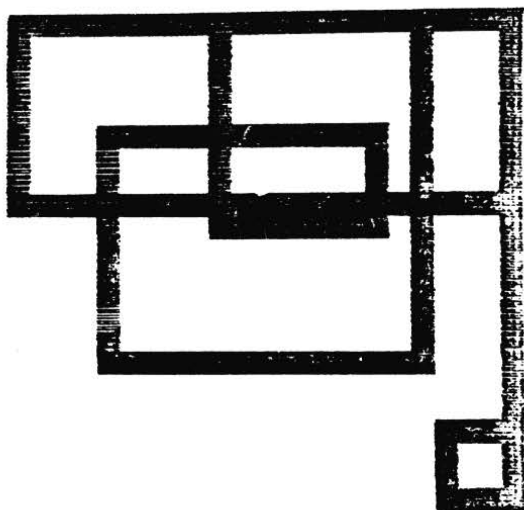
La macchia non viene cancellata dopo il movimento, perciò questo programma ti permette di disegnare la tua figura. Se tu disegni con lo stesso colore dello schermo, è come se lo cancellassi. Poiché il Commodore 64 non ha una funzione di ripetizione automatica, se tieni premuto un tasto, è necessario premere e lasciare il tasto ogni volta che vuoi muovere la macchia.

```
10 REM **DISEGNO**
20 PRINT CHR$(147)
30 LET R=0
40 LET C=0
50 LET CL=0
55 POKE 55296+C+R*40,CL
60 POKE 1024+C+R*40,160
70 GET A$
80 IF A$="" THEN GOTO 70
90 IF A$="5" THEN LET C=C-1
100 IF C<=0 THEN LET C=0
110 IF A$="6" THEN LET R=R+1
120 IF R>=24 THEN LET R=24
130 IF A$="7" THEN LET R=R-1
140 IF R<=0 THEN R=0
150 IF A$="8" THEN LET C=C+1
160 IF C>=39 THEN LET C=39
170 IF A$="9" THEN LET CL=CL+1
180 IF CL>=16 THEN LET CL=0
190 GOTO 55
```

Programma 11 – Un disegno con una macchia

- ESEGUI IL PROGRAMMA 11 -

Un'esecuzione (RUN) tipica.



Se cambi il codice del carattere "spazio" (160) nella linea 60 con un altro carattere, puoi far variare i tuoi disegni. Le linee 55 e 60 ritornano indietro alla mappa dello schermo e alla mappa del colore; C è il numero di colonna (0 a 39) ed R è il numero di riga (da 0 a 24).

Esempio 5

Come muovere una macchia
Sviluppare giochi complessi come la "battaglia navale" va al di là degli scopi di questo corso, in quanto sarebbero necessari i più alti livelli di capacità di programmazione. Comunque, il programma sviluppato dall'esempio 4 mostra come produrre il movimento bidimensionale di una figura. Osserva come differisce dal programma 11, per far sì che la macchia cancelli la propria coda. E' semplice farlo: basta pulire lo schermo prima di tornare indietro e disegnare la posizione successiva.

```
10 REM **PROGRAMMA CHE MUOVE UN PUNTO**
20 PRINT CHR$(147)
30 LET R=0
40 LET C=0
50 LET CL=0
55 POKE 55296+C+R*40,CL
60 POKE 1024+C+R*40,160
70 GET A$
80 IF A$="" THEN GOTO 55
90 IF A$="5" THEN LET C=C-1
100 IF C<=0 THEN LET C=0
110 IF A$="6" THEN LET R=R+1
120 IF R>=24 THEN LET R=24
130 IF A$="7" THEN LET R=R-1
140 IF R<=0 THEN R=0
150 IF A$="8" THEN LET C=C+1
160 IF C>=39 THEN LET C=39
170 IF A$="9" THEN LET CL=CL+1
180 IF CL>=16 THEN LET CL=0
190 PRINT CHR$(147)
200 GOTO 55
```

Programma 12 – Come muovere una macchia.

- ESEGUI IL PROGRAMMA 12 -

Esegui il programma 12 e vedi come puoi muovere la macchina in ogni punto all'interno dell'area prestabilita dello schermo, usando i tasti 5, 6, 7 e 8. Il tasto 9 ti permette di cambiare colore.

7.8 Disegnare una bandiera

Il programma seguente disegna la bandiera italiana, tutta colorata. La bandiera è costituita da tre strisce verticali: da sinistra a destra sono verde, bianco e rosso. Il margine è posto a 0 (nero), per contrasto, e poi il ciclo da 100 a 180 costruisce la bandiera dall'alto al basso, riempiendo una riga di caratteri per volta; la linea 100 è:

100 FOR R=0 TO 24

perché abbiamo 25 linee nella bandiera. La linea 110 è

110 FOR C=0 TO 2

perché abbiamo tre strisce nella bandiera. La linea 130 costruisce una linea orizzontale alla volta. Abbiamo bisogno del RESTORE alla linea 170 per non ripetere i dati per ogni linea.

```
10 REM *****************
20 REM **TRICOLORE VERTICALE**
40 POKE 53280,0
50 POKE 53281,0
60 PRINT CHR$(147)
70 FOR X=1024 TO 2023
80 POKE X,160
90 NEXT X
100 FOR R=0 TO 24
110 FOR C=0 TO 2
120 READ CL
130 FOR X=55296+13*C+40*R TO 55309+13*C+40*R
140 POKE X,CL
150 NEXT X
160 NEXT C
170 RESTORE
180 NEXT R
```

```
190 GOTO 190
200 DATA 5,1,2
```

Programma 13 – Bandiera tricolore verticale.

- ESEGUI IL PROGRAMMA 13 -

Fai eseguire (RUN) il programma

DAV2

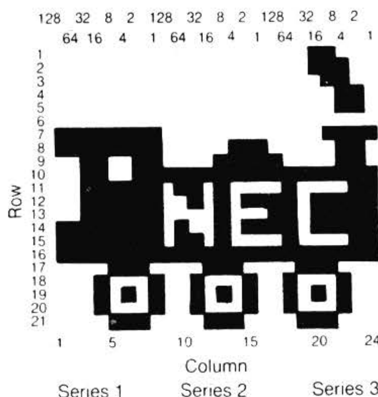
1. Cambia il programma 13 per disegnare la bandiera belga (strisce colorate nera/gialla/rossa). Avrai bisogno di cambiare il colore del margine perché una delle strisce della bandiera è nera.
2. Cambia il programma 13 in modo da disegnare la bandiera francese (strisce verticali blu/bianca/rossa).

Esercizio 1

Abbiamo visto come disegnare una bandiera fatta di 3 strisce verticali, prova a scrivere un programma per tracciare una bandiera di tre strisce orizzontali (potresti provare ad esempio la bandiera olandese: la striscia in alto è rossa, quella in mezzo è bianca e quella in basso è blu).

7.9 La grafica con i folletti (Sprite)

Non entreremo nei dettagli su come definire il video nella grafica ad alta risoluzione. Sarà sufficiente lavorare con degli esempi.
Nella seguente illustrazione,



abbiamo una griglia di 24 colonne e 21 righe, cui possiamo costruire il nostro disegno dettagliato. La griglia è divisa in tre sezioni verticali, chiamate serie 1, serie 2 e serie 3, rispettivamente. Ogni colonna è numerata in potenze di 2 (1, 2, 4, 8, 16, 32, 64 e 128) da destra a sinistra, all'interno di ogni serie. Usando questo schema costruisci un disegno come quello del trenino NEC che fa il giro del sud: Napoli–Enna–Catania.

```
10 REM **TRENO AD ALTA RISOLUZIONE***
20 PRINT CHR$(147)
30 LET HR=53248
40 POKE 53281,8
50 POKE HR+21,4
60 POKE 2042,13
70 POKE HR+41,2
80 FOR T=0 TO 62
90 READ D
100 POKE 832+T,D
110 NEXT T
120 POKE HR+23,4
130 POKE HR+29,4
140 REM **MUOVE TRENO**
150 FOR C=0 TO 255
160 POKE HR+4,C
170 POKE HR+5,150
180 NEXT C
190 GOTO 140
200 DATA 0,0,24,0,0,28,0,0,12,0,0,6,0,0,6,0,0,0
210 DATA 255,0,15,255,7,6,51,15,134,
220 DATA 51,255,255,63,104,67,63,43,223
230 DATA 63,8,223,255,75,223,255,104,67
240 DATA 255,255,255,14,28,56,17,34,68
250 DATA 21,42,84,17,34,68,14,28,56
```

Programma 14 – Il Trenino NEC

- ESEGUI IL PROGRAMMA 14 -

I dati nelle linee da 200 a 250 descrivono il disegno del treno in ordine di riga da 1 a 21, andando lungo le serie da 1 a 3 per ogni riga.
Il numero di DATA per ogni serie, in ogni riga, è ottenuto aggiungendo i numeri per ogni quadrato colorato all'interno.

Linea 20 Pulisce lo schermo
Linea 30 La locazione di memoria 53248 è l'indirizzo di partenza che controlla il video.
Linea 40 Colora lo schermo (sfondo) in arancione
Linea 50 Il registro 21 accende o spegne il disegno del folletto/i; 4 (numero binario 100) significa attivare il folletto 2 (il Commodore 64 può gestire simultaneamente 8 folletti).
Linea 60 Dice al computer di leggere i dati del folletto 2024 che corrisponde al folletto 2.
Linea 70 Il registro 41 riguarda il colore del folletto 2 – questa linea fa tracciare il colore rosso (colore 2).
80 fino a 100 leggono i 62 elementi di DATA (21 linee x 3 serie) e li immagazzinano nelle 832 locazioni successive.
Linee 120 e 130 "Espandono" il disegno del folletto nella direzione X (sinistra – destra) e nella direzione Y (sopra–sotto): in questo modo il disegno del folletto diventa più grande e più facile da usare.
Linee 150 fino a 180 muovono il disegno del folletto: 160 lo muove da sinistra a destra; 170 mantiene il disegno sullo stesso livello nella direzione Y (perché un treno non va per aria!)

Ora fai eseguire (RUN) il programma 14. Vedrai il treno muoversi sullo schermo da sinistra a destra. Se tracci un disegno, come quello del treno, su un foglio di carta quadrata e poi lo traduci in dati potrai usare il programma 14 per animare il disegno del tuo folletto.

7.10 Suoni nei programmi

Il Commodore 64 è molto versatile nella generazione di suoni, può emettere dei semplici toni e degli effetti sonori ma anche armonie complesse. Il suono viene riprodotto attraverso l'altoparlante della televisione che usi come schermo. Il capitolo 7 del manuale per l'utente riporta le tabelle di dati.
Nel Commodore 64, le istruzioni per la generazione dei suoni, come quelle grafiche, sono molto complicate. Parleremo solo di una delle tre voci disponibili. Per ogni nota, dovremo dare al computer istruzioni che indichino:
- il volume della nota,
- quanto rapidamente cresce e cade la nota dal volume di livello più alto,
- le forme dell'onda (tipo di suono prodotto),
- quanto a lungo deve essere mantenuta la nota,
- le lunghezze della nota e
- i due valori, chiamati alta e bassa frequenza della nota.

Ognuno di questi elementi è controllato da una POKE ad una locazione di memoria numerata. Nel nostro programma daremo nomi significativi a queste locazioni di memoria, per rendere il listing un po' più facile da seguire. Noi ci occuperemo della VOCE 1. In questo caso il volume è controllato dalla locazione 54296, l'onda dalla locazione 54276, l'attacco e il decadimento più in basso). La linea 120 pone a livello medio il valore di sostegno; il ciclo fittizio delle linee 130 e 140 determinano la lunghezza delle note. Più grande è il numero contenuto in riga 130 e più lunga sarà la nota.
Le linee 150 e 160 leggono i valorii delle frequenze alte e basse delle note di cui abbiamo bisogno; la linea 170, infine, controlla che tutti i dati vengano letti. I dati del programma si riferiscono alla quarta ottava della scala di DO.

```
10 REM **SCALA MUSICALE**
20 REM *****************
30 LET VOL=54296
40 LET WAV=54276
50 LET ATD=54277
60 LET LFRQ=54272
70 LET HFRQ=54273
80 LET SUS=54278
90 POKE VOL,15
100 POKE WAV,17
110 POKE ATD,129
120 POKE SUS,64
130 FOR D=1 TO 250
140 NEXT D
150 READ H
160 READ L
170 IF H<0 THEN GOTO 220
180 POKE LFRQ,L
190 POKE HFRQ,H
200 POKE WAV,0
210 GOTO 100
220 REM **FINE**
230 POKE LFRQ,0
240 POKE HFRQ,0
250 END
260 REM **QUARTA OTTAVA DELLA SCALA DI DO**
270 DATA 17,37
280 DATA 19,63
290 DATA 21,154
300 DATA 22,227
```

```
310 DATA 25,177
320 DATA 28,214
330 DATA 32,94
340 DATA 34,75
350 DATA -1,-1

READY.
```

Programma 15 - Scala di DO.

- ESEGUI L PROGRAMMA 15 -

Alza il volume del televisore, dai il RUN ed ascolta la scala: DO RE MI FA SOL LA SI DO

Esercizio 2

Cerca di trovare i dati della quarta ottava della scala di SOL usando il programma 15 ora, per sentirci un motivetto, abbiamo bisogno di cambiare la lunghezza delle note; per fare ciò basta alterare la lunghezza del ciclo alle righe 190 e 200. Per ascoltare un motivo con volume, attacco/decadimento e sostegno costanti dobbiamo basarci su tre dati, l'alta frequenza, la bassa frequenza e la durata.

Cambiando la linea 190 è possibile variare la velocità del motivo.

```
190 FOR T=1 TO D/2
```

Darà bevi note,

```
190 FOR T=1 TO D*2
```

Potrà essere, invece, utile per correggere possibili errori. Data la velocità dimezzata.

```
10 REM ******************
15 REM **INNO NAZIONALE**
20 REM ******************
30 LET VOL=54296
40 LET WAV=54276
```

```
50 LET ATD=54277
60 LET LFRQ=54272
70 LET HFRQ=54273
80 LET SUS=54278
90 POKE VOL,15
100 POKE WAV,33
110 POKE ATD,129
120 POKE SUS,128
130 READ H
140 READ L
150 READ D
160 IF H<0 THEN GOTO 240
170 POKE LFRQ,L
180 POKE HFRQ,H
190 FOR T=1 TO D
200 NEXT T
210 POKE LFRQ,0
220 POKE HFRQ,0
230 GOTO 130
240 END
250 REM **FREQUENZE E DURATA DELLE NOTE**
270 DATA 19,137,300
280 DATA 19,137,300
290 DATA 21,237,150
300 DATA 19,137,450
310 DATA 32,218,300
320 DATA 32,218,300
330 DATA 34,206,150
340 DATA 32,218,450
350 DATA 32,218,300
360 DATA 39,17,300
390 DATA 34,206,150
400 DATA 32,218,450
410 DATA 29,69,300
420 DATA 32,218,300
430 DATA 29,69,150
440 DATA 26,19,450
450 DATA 32,218,300
460 DATA 32,218,300
470 DATA 24,156,600
480 DATA 26,19,150
490 DATA 29,69,150
500 DATA 26,19,150
```

```
510 DATA 24,156,150
520 DATA 21,237,450
530 DATA 26,19,300
540 DATA 24,156,300
550 DATA 26,19,150
560 DATA 29,69,450
570 DATA 19,137,300
580 DATA 32,218,600
590 DATA 34,206,150
600 DATA -1,-1,-1
```

Programma 16 - L'inno nazionale.

- ESEGUI IL PROGRAMMA 16 -

```
10 REM **DARBY KELLY**
20 REM ***************
30 LET VOL=54296
40 LET WAV=54276
50 LET ATD=54277
60 LET LFRQ=54272
70 LET HFRQ=54273
80 LET SUS=54278
90 POKE VOL,15
100 POKE WAV,33
110 POKE ATD,136
120 POKE SUS,128
130 READ H
140 READ L
150 READ D
160 IF H<0 THEN GOTO 240
170 POKE LFRQ,L
180 POKE HFRQ,H
190 FOR T=1 TO D
200 NEXT T
210 POKE LFRQ,0
220 POKE HFRQ,0
230 GOTO 130
240 END
250 REM **DARBY KELLY**
260 REM **FREQUENZE E DURATA**
270 DATA 19,63,120,25,177,240
280 DATA 32,94,120,32,94,240
290 DATA 19,63,120,25,177,240
```

```
300 DATA 32,94,120,32,94,240
310 DATA 19,63,120,25,177,240
320 DATA 32,94,120,32,94,80
330 DATA 28,214,80,32,94,80
340 DATA 34,75,240,28,214,120
350 DATA 28,214,240,19,63,120
360 DATA 25,177,240,32,94,120
390 DATA 32,94,80,28,214,,80
400 DATA 25,177,80,21,154,240
410 DATA 34,75,120,34,75,240
420 DATA 21,154,120,19,63,240
430 DATA 32,94,120,32,94,240
440 DATA 28,214,120,28,214,240
450 DATA 25,177,120,25,177,240
460 DATA 23,94,120,28,214,240
470 DATA 19,63,120,19,63,240
480 DATA 32,94,120,28,214,240
490 DATA 19,63,120,19,63,240
500 DATA 32,94,120,28,214,240
510 DATA 38,126,120,38,126,80
520 DATA 36,85,80,38,126,80
530 DATA 38,126,240,19,63,120
540 DATA 19,63,960,34,75,120
550 DATA 32,94,240,28,214,120
560 DATA 25,177,240,24,63,120
570 DATA 21,154,240,25,177,120
580 DATA 25,177,80,21,154,80
590 DATA 21,154,80,19,63,240
600 DATA 32,94,120,32,94,240
610 DATA 28,214,120,28,214,240
620 DATA 25,177,120,25,177,240
630 DATA 32,94,120,28,214,240
640 DATA 19,63,120,19,63,240
650 DATA 32,94,120,28,214,240
660 DATA 19,63,120,19,63,240
670 DATA 32,94,120,28,214,240
680 DATA 38,126,120,38,126,80
690 DATA 36,85,80,38,126,80
700 DATA 38,126,240,19,63,120
710 DATA 19,63,960,34,75,120
720 DATA 32,94,240,28,214,120
730 DATA 25,177,240,24,63,120
740 DATA 21,154,240,25,277,120
```

```
750 DATA 25,177,80,24,63,80
760 DATA 21,154,80,19,63,240
770 DATA 32,94,120,32,94,240
780 DATA 28,214,120,28,214,240
790 DATA 25,177,120,25,177,140
800 DATA -1,-1,-1
```

Programma 17 – Darby Kelly.

- ESEGUI IL PROGRAMMA 17 -

Come vedi, sul Commodore 64, possono essere suonati motivi abbastanza complessi, anche usando una sola voce. Se conosci un po' di teoria musicale, non dovresti trovare difficoltà a scrivere programmi musicali.
Altrimenti, su un qualsiasi libro che contenga motivi semplici, potrai trovare dei motivi idonei, con cui cominciare. Scrivi il nome di ogni nota (DO,RE,MI) sotto ogni nota del pezzo musicale, e poi controlla il pentagramma sul Manuale per l'utente del tuo C64, in modo da trovare i valori dell'alta e della bassa frequenza. Decidi quali valori "standard" dare alla durata delle note (per es. 100 è la semiminima) e poi stabilisci il valore di ogni nota nel tuo motivo.

7.11 Effetti sonori nei programmi

Una volta scritto un pezzo musicale corretto, il Commodore 64 può produrre un'ampia varietà di effetti sonori nel programma. Conoscerai di certo le esplosioni o i fischi dei vecchi flipper o di altri giochi disponibili in commercio.
Torniamo al nostro vecchio amico, l'autobus che si muove. Nel seguente programma, sono state aggiunte le linee da 72 a 75 e da 161 a 168, per favorire un suono a due toni. Questo rallenta il programma, perciò le linee 150 e 160 non sono più necessarie.

```
10 REM ***************
20 REM **MOVIMENTO DEL BUS**
30 REM ***************
40 LET B$(1)="{RED}{RVS ON}{CBM D}{
   5 CBM F}{RVS OFF}"
50 LET B$(2)="{RED}{RVS ON}{7 SPC}{
   RVS OFF}"
60 LET B$(3)="{BLACK} O{3 SPC}O"
```

```
70 LET C$="{38 SPC}"
72 POKE 54296,15
73 POKE 54276,17
74 POKE 54277,129
75 POKE 54278,128
80 FOR M=1 TO 32
90 PRINT"{CLR}{5 CUR.GIU}"
100 PRINT"{BLACK}";
105 PRINT"{38 CBM U}"
110 PRINT"{5 CUR.SU}"
120 FOR K=1 TO 3
130 PRINT LEFT$(C$,M);B$(K)
140 NEXT K
161 POKE 54273,34
162 POKE 54272,75
163 FOR T=1 TO 40
164 NEXT T
165 POKE 54273,17
166 POKE 54272,37
167 FOR T=1 TO 40
168 NEXT T
170 NEXT M
180 PRINT"{SH Z}"
```

Programma 18 – L'autobus con suoni

- ESEGUI IL PROGRAMMA 18 -

Il programma suona due DO spaziati di un'ottava alternativamente, mentre l'autobus si muove, simulando il rumore del motore.

DAV3

Cambia il programma 18 per rendere rosso il margine, giallo lo sfondo e verde l'autobus.

Compito 7

1. Prendi il programma 8, il test sulla tabellina, ed aggiungi effetti sonori adatti alla versione a colori:
   (a) per accompagnare le istruzioni iniziali
   (b) per accompagnare le risposte corrette
   (c) per accompagnare le risposte errate
   (d) per accompagnare i punteggi finali e le riesecuzioni.

Ti diamo dei suggerimenti: stabilisci prima esattamente dove si trova, nel programma, ognuna delle 4 funzioni indicate, decidi poi l'effetto sonoro adatto (es. una scala crescente per le risposte esatte, una decrescente per gli errori, ecc.)

2. Scegli una semplice bandiera (diversa da quelle di questa unità) e scrivi un programma per disegnarla, a colori.

Obiettivi dell'unità 7

Quando hai finito questa unità controlla se sei capace di:

Cambiare il colore del margine dello schermo ☐
Cambiare il colore dello sfondo ☐
Cambiare il colore dell'inchiostro ☐
Descrivere come la combinazione di luci colorate forma altri colori ☐
Aggiungere colori ai programmi per migliorare gli output ☐
Produrre semplici miglioramenti grafici ☐
Usare un programma con la grafica dei folletti ☐
Suonare i motivi ☐
Aggiungere effetti sonori ☐

Risposte alle DAV e agli esercizi

DAV1

(a) margine giallo
(b) sfondo blu
(c) inchiostro porpora

DAV2

1. Bandiera belga
   cambiamenti al programma 13
   40 POKE 53280,1
   50 POKE 53281,1
   200 DATA 0,7,2
2. Bandiera francese
   cambiamenti al programma 13:
   200 DATA 6,1,2
   (40 e 50 vanno bene)

Esercizio 1

Bandiera tricolore orizzontale

```
10 REM ****************
20 REM **TRICOLORE ORIZZONTALE**
30 REM ****************
40 POKE 53280,0
50 POKE 53281,0
60 PRINT CHR$(147)
70 FOR X=1024 TO 2023
80 POKE X,160
90 NEXT X
100 FOR C=0 TO 2
110 READ CL
120 FOR X=55296+320*C TO 55615+320*C
130 POKE X,CL
140 NEXT X
150 NEXT C
160 GOTO 160
170 DATA 2,1,6
```

Programma 19 – Bandiera tricolore orizzontale

- ESEGUI IL PROGRAMMA 19 -

Esercizio 2

Scala musicale di SOL

```
10 REM **SCALA MUSICALE**
20 REM ******************
30 LET VOL=54296
40 LET WAV=54276
50 LET ATD=54277
60 LET LFRQ=54272
70 LET KFRQ=54273
80 LET SUS=54278
90 POKE VOL,15
100 POKE WAV,17
```

```
110 POKE ATD,29
120 POKE SUS,64
130 FOR D=1 TO 250
140 NEXT D
150 READ H
160 READ L
170 IF H<0 THEN GOTO 220
180 POKE LFRQ,L
190 POKE HFRQ,H
200 POKE WAV,0
210 GOTO 100
220 REM **FINE**
230 POKE LFRQ,0
240 POKE HFRQ,0
250 END
260 REM **QUARTA OTTAVA DI SOL**
270 DATA 25,177
280 DATA 28,214
290 DATA 32,94
300 DATA 34,75
310 DATA 38,126
320 DATA 43,52
330 DATA 48,127
340 DATA 51,97
350 DATA -1,-1
```

Programma 20 – Scala di SOL

- ESEGUI IL PROGRAMMA 20 -

DAV3

Cambiamenti al programma autobus (programma 18)
linee 40 e 50 – Il primo elemento tra virgolette dovrebbe essere CTRL6
aggiunti 35 POKE 53280,2 per ottenere il margine rosso
aggiungi 36 POKE 53281,7 per ottenere lo sfondo giallo.

UNITÀ 8
Manipolazione di numeri

8.1 Introduzione

Abbiamo dimostrato che i computer non sono dei mangiatori di numeri, ma in questa unità daremo più da vicino un'occhiata alla loro capacità aritmetica. Ci concentreremo su un'aritmetica molto semplice, perciò non temere, non incontrerai grosse difficoltà.

Continueremo con il nostro approccio del "vediamo cosa succede", e del "facciamoci dire dalla macchina cosa sta facendo".
Speriamo che anche tu adotterai lo stesso approccio con la macchina che usi.

8.2 Medie e media aritmetica

"Quanto tempo hai studiato ogni unità del corso?" "Beh, in media 3 ore". Se ti chiediamo quanto tempo impieghi nel percorrere un tragitto regolare, per esempio per andare al lavoro, ci rispondi in modo simile. Chi segue lo sport usa termini come media di goal o media di battute; gli atlanti geografici sono pieni di disegni che illustrano la media delle precipitazioni nei vari mesi dell'anno. Parliamo di valore medio in un test o di età media di un gruppo di persone, ecc.

Se volessimo calcolare la media o la media aritmetica dell'età di un gruppo particolare, sommeremmo le età dei membri del gruppo e divideremmo quella somma per il numero totale di persone nel gruppo. Per trovare una media aritmetica si sommano gli addendi e poi si divide la somma per il numero degli addendi.

Esempio 1

Trova la media aritmetica del seguente insieme di numeri:

6, 7, 2, 5, 4, 4, 9, 8.

Soluzione

La somma è $6+7+2+5+4+4+9+8 = 45$
Siccome addendi sono 8 la loro media aritmetica sarà $45/8 = 5.625$.

DAV1

Trova la media aritmetica del seguente insieme di numeri:

8, 4, 2, 6, 1, 7, 6, 1, 4.

Media aritmetica

Esempio 2

Progetta un algoritmo e scrivi un programma per trovare la media aritmetica dei numeri immagazzinati nelle istruzioni DATA che seguono:

```
900 DATA 56,47,52,65,24,34,59,37,49,66
910 DATA 38,24,62,76,31,47,66,61,74,45
920 DATA 66,44,55,67,36,56,54,54,50,43
930 DATA 18,83,23,79,29,-9999
```

Soluzione

Esprimeremo per prima cosa l'algoritmo in forma descrittiva:

1. Inizio
2. Poni il contatore a 1
3. Poni la somma a 0
4. Immetti il valore successivo
5. Se il valore = –9999 allora vai a 9, altrimenti procedi a 6
6. Aggiungi il valore alla somma
7. Aggiungi 1 al contatore
8. Vai a 4
9. Sottrai 1 al totale del contatore
10. Calcola la media = somma/totale
11 Emetti la media

Figura 1 - Algoritmo descrittivo per il calcolo delle medie aritmetiche.

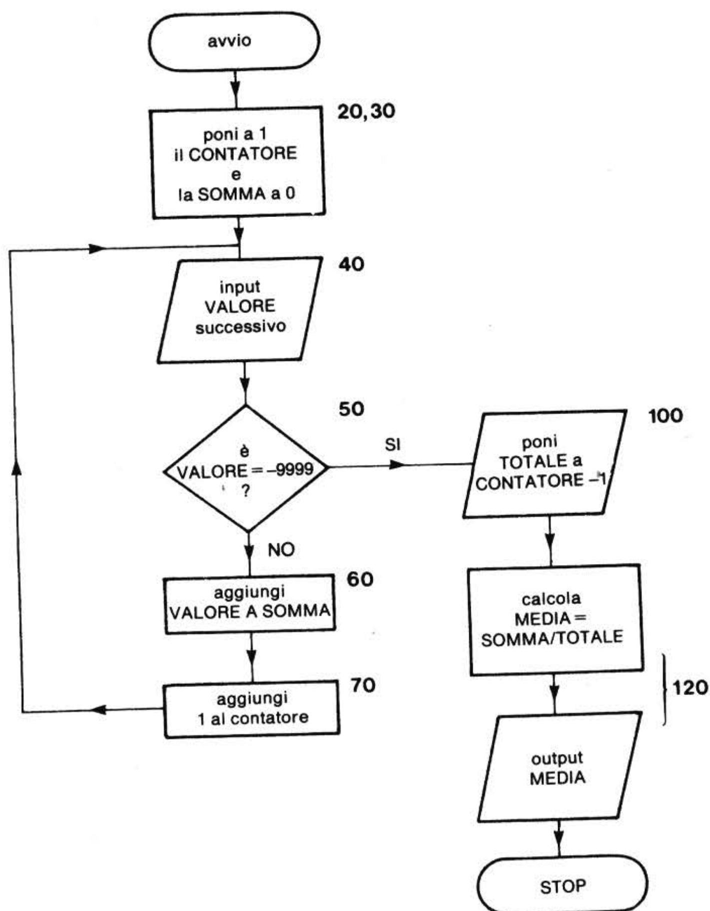


Figura 2 — Diagramma di flusso per la media aritmetica.

```
10 REM **MEDIA ARITMETICA**
15 PRINT CHR$(147)
20 LET C=1
30 LET S=0
35 REM **CICLO SOMMA**
40 READ M
50 IF M=-9999 THEN 95
60 LET S=S+M
```

```
70 LET C=C+1
80 GOTO 35
90 REM *************
95 REM **CONTEGGIO**
100 LET N=C-1
110 REM ************
115 PRINT"PROGRAMMA MEDIA ARITMETICA"
120 PRINT"MEDIA=";S/N
130 REM ************
900 DATA 56,47,52,65,24,34,59,37,49,66
910 DATA 38,24,62,76,31,47,66,61,74,45
920 DATA 66,44,55,67,36,56,54,54,50,43
930 DATA 18,83,23,79,29,-9999
```

Programma 1 - media aritmetica.

```
RUN
PROGRAMMA MEDIA ARITMETICA
MEDIA= 50.5714286

READY
```

- ESEGUI IL PROGRAMMA 1 -

Esercizio 1

Scrivi un programma per trovare la lunghezza media delle parole nelle istruzioni DATA dell'esempio 2 dell'unità 5. Ricordi? La filastrocca sulla "pigrizia". Inserisci una routine all'interno del programma 1 dell'unità 5, che già trova la lunghezza delle parole.

Esercizio 2

Scrivi un programma per trovare il punteggio medio in un esperimento simulato di lancio di una moneta per 100 volte. Fallo girare 30 volte per vedere quale variazione di risultati ottieni.

Simulazione

Il punteggio medio atteso, quando lanci un dado per molte volte, è:

$$\frac{1+2+3+4+5+6}{6} = \frac{21}{6} = 3.5$$

Comunque nelle nostre esecuzioni dell'esercizio 2, abbiamo ottenuto esattamente una volta 3.5 con valori che vanno da 3.24 a 3.76. L'esercizio 2 riguardava i risultati di 3.000 lanci (30x100), perciò potresti giustamente chiederti: "Il generatore di numeri casuali sta mentendo?" (noi lo avevamo chiamato "pseudo", infatti!). Di quanti esperimenti avremo bisogno per convincerci se mente o no?
Per indagare a fondo la questione, dovremmo entrare nella teoria statistica e ciò va oltre gli scopi di questo corso ma possiamo trovare almeno la media di queste medie.
Tutto ciò che dobbiamo fare è prendere i dati delle trenta esecuzioni del programma:

3.56, 3.47, 3.52, 3.65,...

ed immetterli nelle istruzioni DATA del programma 1. Otterremo la media delle medie.
Noterai che qui sotto abbiamo immesso solo le parti decimali dei numeri per rendere l'immissione dei nostri dati un poco più semplice, per es. 56 invece di 3.56 (possiamo far ciò perché tutti i numeri sono 3 più qualcosa)

```
130 REM **********
900 DATA 56,47,52,65,24,34,59,37,49,66
910 DATA 38,24,62,76,31,47,66,61,74,45
920 DATA 66,44,55,67,36,56,54,54,50,43
930 DATA -9999
940 END
```

Figura 3 - Istruzioni DATA

Perciò, la media totale di 3000 lanci è 3.51 ed altri decimali che qui abbiamo tolto: tutto questo sembra molto più convincente!

Per riassumere

"Simulazione" è una parola forse un po' troppo grossa per indicare quello che abbiamo appena fatto. Comunque volevamo sottolineare che possiamo simulare un'attività della vita reale senza addentrarci a fondo nella statistica. Non avremmo potuto lanciare un dado per 3.000 volte in una classe scolastica ma con un computer possiamo raccogliere e trattare dati abbastanza rapidamente.
Se questo esercizio ha stimolato la tua curiosità, allora prova a passare all'esercizio successivo.

Esercizio 3

Scrivi un programma per trovare il punteggio medio in un esperimento simulato del lancio di due dadi per 100 volte.
Quale punteggio medio ti aspetti in questo caso e in esperimenti con 3, 4...dadi? Le tue aspettative sono avvalorate dagli esperimenti?

8.3 Intervallo di variazione

Discutendo i risultati dell'esercizio 2, nella pagina precedente, abbiamo usato in modo abbastanza naturale l'idea di intervallo di variazione. Abbiamo detto che i valori variano da 3.24 a 3.76. Questo processo, infatti, implica che vadano trovati i valori più alti e più bassi dell'insieme.

Esercizio 3

Progetta un algoritmo e scrivi una routine per trovare i valori minimo e massimo dei numeri contenuti nelle istruzioni DATA nell'esempio 2. Aggiungi questa routine al programma per trovare la media aritmetica indicata nella soluzione all'esempio 2 (se ti senti abbastanza sicuro, affronta questo esempio come esercizio, prima di procedere con la soluzione).

Soluzione

Potresti avere la sensazione di essere tornato indietro, all'unità 4, quella in cui abbiamo trovato il membro inferiore di una lista. Potremmo usare di nuovo lo stesso approccio ma, in questo caso, per prima cosa dovremmo porre i dati

in forma di lista e poi riorganizzare la lista una seconda volta con una routine di scambio. Questo è un lavoro molto impegnativo perciò ricorreremo ad un approccio più breve: provare a cercare i valori più alti e più bassi mentre vengono letti i dati.
Sappiamo già come si legge all'interno di DATA (linee 10–50 nel programma 1) ma cosa farcene di ogni elemento letto?

- Per prima cosa creiamo due depositi:
  M per il valore superiore
  B per il valore inferiore

- Poi leggiamo il primo valore e lo mettiamo sia in B che in M. Dopo tutta quest'operazione, lo stesso valore è sia superiore che inferiore!

- Quindi leggiamo ogni valore e se è maggiore di quello in M, lo mettiamo in M oppure, se è minore di quello in B, lo mettiamo in B. Se non è maggiore di M, o non è minore di B, passiamo al valore successivo.

Una soluzione descrittiva, pertanto, potrebbe essere:

Descrizione

1. Routine di avvio
2. Se il contatore = 1 allora scrivi il valore nel deposito superiore e vai a 6, altrimenti procedi a 3.
3. Se il valore > superiore, allora scrivi il valore in superiore e vai a 6, altrimenti vai a 4.
4. Se il valore > = inferiore, allora vai a 6, altrimenti vai a 5.
5. Scrivi il valore in inferiore
6. Fine della routine.

È un diagramma di flusso:

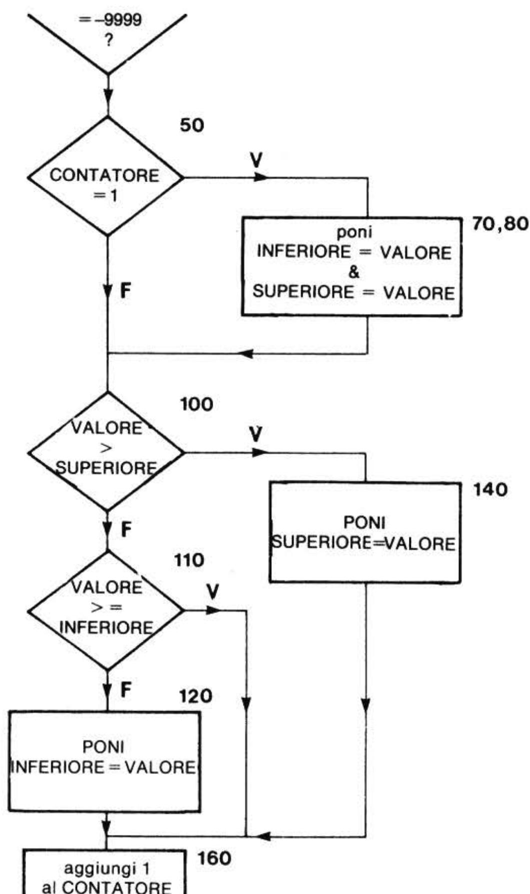


Figura 4 - Diagramma di flusso.

Quale trovi sia più facile da seguire?

Un algoritmo descrittivo può risultare piuttosto confuso, se in un programma ci sono parecchie diramazioni. Un diagramma di flusso, bidimensionale, può risultare più utile. E' un problema di scelta personale, giudica da te!

```
10 REM **MASSIMO E MINIMO**
15 PRINT CHR$(147)
20 LET C=1
25 LET S=0
28 REM **LEGGE UN VALORE**
30 READ M
40 IF M=-9999 THEN 185
50 IF C>1 THEN 95
60 REM *************
70 LET B=M
80 LET T=M
90 REM *************^
95 REM **CONTROLLO PER MASSIMO/MINIMO**
100 IF M>T THEN 135
110 IF M>=B THEN 155
120 LET B=M
130 GOTO 155
135 REM **NUOVO VALORE MASSIMO**
140 LET T=M
150 REM *************
155 REM **TOTALE PER LA MEDIA**
160 LET C=C+1
165 LET S=S+M
170 GOTO 28
180 REM *************
185 REM **OUTPUT**
190 PRINT"MASSIMO=";T,"MINIMO=";B:PRINT
200 REM *************
210 LET N=C-1
220 PRINT"MEDIA=";S/N
230 REM *************
900 DATA 56,47,52,65,24,34,59,37,49,66
910 DATA 38,24,62,76,31,47,66,61,74,45
920 DATA 66,44,55,67,36,56,54,54,50,43
930 DATA 18,83,23,79,-9999
```

(linea 70) il programma visita questo punto solo la prima volta nel ciclo, quando C = 1

(linee 100–110) a questo punto vengono prese le decisioni

Programma 2

```
RUN
MASSIMO= 83           MINIMO= 18

MEDIA= 51.2058824

READY
```

- ESEGUI IL PROGRAMMA 2 -

Esercizio 4

Scrivi un programma per tracciare una tabella di frequenza per i dati del programma 2, usando le categorie:

0–9, 10–19, 20–29,...90–99

Suggerimenti

Potresti usare una lista di punteggi:
S(0), S(10), S(20),...S(100)
e, per ogni valore letto, controllare se questo sia minore rispetto al superiore della seconda lista (10), minore rispetto al superiore della terza lista (20), finché trovi

VALORE < K vero

allora incrementa S(K–10). Questo è l'approccio che abbiamo già usato (controlla la risposta alla fine del capitolo).

8.4 Numeri e ancora numeri

Abbiamo cercato in questo corso di evitare qualsiasi aritmetica che non fosse completamente semplice. Continueremo a farlo. Daremo comunque un breve sguardo alle capacità aritmetiche del computer. Se ti spaventi alla vista di queste poche pagine, e decidi di passare direttamente al paragrafo successivo non perderai informazioni vitali per la programmazione. Speriamo però che tu voglia fare lo stesso un tentativo. Il Commodore 64 certamente si farà una scorpacciata di aritmetica.
Qui c'é un semplice programma che calcola i quadrati da 1 a 10 (linea 50), i cubi (linea 60) i reciproci (linea 70) e ne tabula i risultati.

```
1 REM **TABULAZIONE DI QUADRATI,CUBI E
2 REM RECIPROCHI DEI PRIMI DIECI
3 REM NUMERI NATURALI**
5 PRINT CHR$(147)
```

```
10 PRINT" N";TAB(7);"N*N";TAB(13);"N*N*N";TAB(21)
;"1/N";
20 PRINT
25 REM **CICLO DI CALCOLO**
30 FOR I=1 TO 10
35 REM **NUMERO**
40 LET S=1*I
45 REM **QUADRATO**
50 LET S=I*I
55 REM **CUBO**
60 LET C=I*I*I
65 REM **RECIPROCO**
70 LET R=1/I
75 REM **STAMPA**
80 PRINT I;TAB(7);S;TAB(13);C;TAB(21);R
90 NEXT I
95 REM **FINE DEL CICLO**
100 END
```

Programma 3

```
RUN
 N      N*N    N*N*N    1/N
 1       1      1       1
 2       4      8       .5
 3       9      27      .333333333
 4       16     64      .25
 5       25     125     .2
 6       36     216     .166666667
 7       49     343     .142857143
 8       64     512     .125
 9       81     729     .111111111
 10      100    1000    .1

READY
```

- ESEGUI IL PROGRAMMA 3 -

Elevazione a potenza

Sicuramente questo tipo di notazione ti sarà familiare:

$4x4 = 4^2$ (4 al quadrato o elevato alla seconda)
7x7x7 = 7 (7 al cubo o elevato alla terza)
10x10x10x10x10 = 10] (10 elevato alla quinta)

In BASIC l'elevazione a potenza è indicato con il segno ↑. Qualsiasi numero N elevato alla potenza P è scritto come N ↑ P. P è chiamato "esponente" e l'operazione relativa l'elevazione a potenza.
Così, per le potenze negative:

	N	P	N↑P
$\frac{1}{4} \times \frac{1}{4} = \frac{1}{4 \times 4} = 4^{-2}$	4	−2	4↑(−2)
$\frac{1}{7} \times \frac{1}{7} \times \frac{1}{7} = \frac{1}{7 \times 7 \times 7} = 7^{-3}$	7	−3	7(−3)
$\frac{1}{10} \times \frac{1}{10} \times \frac{1}{10} \times \frac{1}{10} \times \frac{1}{10} = \frac{1}{10 \times 10 \times 10 \times 10 \times 10} = 10^{-5}$	10	−5	10↑(−5)

Non affrontiamo qui le potenze fratte (positive e negative).
Possiamo usare le notazione ↑ invece dell'asterisco nel programma 3. Per cui, il programma 3, riscritto, diventa:

```
1 REM **TABULAZIONE DI QUADRATI,CUBI E
2 REM RECIPROCHI DEI PRIMI DIECI
3 REM NUMERI NATURALI**
10 PRINT" N";TAB(4);"N^2";TAB(16);"N^3";TAB(28)
;"N^(-1)":PRINT
20 FOR N=1 TO 10
30 PRINT N;TAB(3);N^2;TAB(15);N^3;TAB(27);N^(-1)
40 NEXT N
50 END
```

Programma 4

Abbiamo inserito qui questo programma per mostrarti che l'operazione di elevazione a potenza non è molto precisa con il Commodore 64. Il paragrafo 8.6, riguardante la rappresentazione di numeri, ti spiegherà il perché.

Sequenze e loro somme

Calcolare gli elementi individuali in una sequenza, o la somma dei primi N elementi, è un lavoro immane senza un computer. Quanto tempo impiegheresti per calcolare il valore degli elementi di

$$\frac{1}{1^2}, \frac{1}{2^2}, \frac{1}{3^2}, \ldots \frac{1}{N^2}?$$

Bene, è molto facile a farsi con il programma 5

```
3 PRINT CHR$(147)
5 PRINT"CALCOLO DI SERIE":PRINT
10 PRINT"{DESTRA}N"," N^(-2)"
15 REM **INIZIO DEL CICLO DI CALCOLO**
20 FOR N=1 TO 10
30 PRINT N,N^(-2)
40 NEXT N
50 END
```

Programma 5

```
RUN
CALCOLO DI SERIE

 N          N^(-2)
 1          1
 2          .25
 3          .111111111
 4          .0625
 5          .04
 6          .0277777778
 7          .0204081633
 8          .015625
 9          .012345679
 10         .01

READY
```

- ESEGUI IL PROGRAMMA 5 -

Esercizio 5

Scrivi un programma per trovare gli elementi della serie:

$1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots$

necessari perché la loro somma superi 2.4.

Esercizio 6

Modifica il programma dell'esercizio 5 per trovare quanti elementi è necessario sommare

$$= 1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \ldots$$

per superare 1.5.

Esercizio 7

I fattoriali sono numeri interessanti. Il 4 fattoriale (e cioè =4x3x2x1) generalmente viene scritto con 4!
Così
7 fattoriale = 7x6x5x4x3x2x1 = 7!
N fattoriale = Nx(N–1)x(N–2)x...x1 = N!
Scrivi un programma per calcolare i fattoriali di ogni intero positivo.

Esercizio 8

Nell'unità 1 abbiamo scritto un programma piuttosto sgraziato per calcolare il guadagno (Y) su un deposito (D) con una percentuale d'interesse composto. Una formula più accurata è la seguente: Y = Dx(1+P/100)T (dove T è il numero di anni dell'investimento).
Scrivi un programma per calcolare il guadagno, usando questa formula, per depositi, percentuali e periodi di tempo differenti.

8.5 "Dry running" ovvero seguire il flusso di un algoritmo

Spesso un programma non lavora bene o non ci soddisfa completamente. Se abbiamo acquisito una certa dimestichezza col computer, possiamo sederci alla macchina fino a quando troviamo l'errore, altrimenti siamo costretti a provare tutto con carta e penna. E' una bella seccatura!
L'espressione inglese che indica lo scorrimento di un algoritmo linea per linea con penna e carta è detto dry running.
Illustreremo questo procedimento guardando il programma 18 che rappresenta la soluzione all'esercizio 5.

```
10 REM **SOMMA DI RECIPROCI**
15 PRINT CHR$(147)
20 LET S=0
30 LET N=1
35 REM **INIZIO DEL CICLO DI CALCOLO**
40 LET S=S+1/N
50 IF S>2.4 THEN 80
60 LET N=N+1
70 GOTO 35
80 REM **OUTPUT**
90 PRINT"SOMMA=";S:PRINT
95 PRINT"IL NUMERO DEGLI ELEMENTI E' ";N
100 END
```

Programma 18 – (Dall'esercizio 5)

```
RUN
SOMMA=2.45 IL NUMERO DEGLI ELEMENTI E' 6
READY
```

Introduciamo un altro concetto, quello di "tracing", che significa trovare e registrare ogni passo, il numero di linea eseguita in quel passo e il valore delle variabili dopo che la linea è stata eseguita. Perciò, per il programma 18 abbiamo bisogno delle intestazioni:

passo No.	linea No.	N	S
1			
2			
ecc.			

Omettiamo dalle linee tracciate quelle che non riguardano le variabili, cioè:

REM (linea 10)
GOTO (linea 70)
PRINT (linee 15 e 90)

A parte queste linee, i passi del programma sono:

passo No.	linea No.	N	S
1	20	0	0
2	30	1	0
3	40	1	1

Passo No.	Linea no.	N	S	
4	50	1	1	
5	60	2	1	Nota l'effetto del GOTO 40 alla linea 60
6	40	2	1.5	
7	50	2	1.5	
8	60	3	1.5	GOTO di nuovo
9	40	3	1.83	
10	50	3	1.83	
11	60	4	1.83	GOTO
12	40	4	2.08	
13	50	4	2.08	
14	60	5	2.08	GOTO
15	40	5	2.28	
16	50	5	2.28	
17	60	6	2.28	GOTO
18	40	6	2.45	

Figura 5 - I passi del programma.

Tracing

Alcuni interpreti BASIC hanno un comando TRACE, il Commodore 64 no, ma non è grave.

Una routine di tracing attentamente progettata da te lavora sicuramente meglio.

DAV2

Fai il "Dry running" del programma che segue, cominciando quando la linea 20 è stata eseguita e continuando fino a quando la condizione espressa nella linea 50 non sia vera. Disegna la tabella con le stesse intestazioni dell'altro esempio.

```
10 REM **SOMMA DI QUADRATI**
15 PRINT CHR$(147)
20 S=0
30 N=1
35 REM **AGGIUNGE IL TERMINE SUCCESSIVO**
40 S=S+N^2
50 IF S>50 THEN 80
60 N=N+1
70 GOTO 35
80 REM **OUTPUT**
85 PRINT"SOMMA DI QUADRATI":PRINT
90 PRINT"SOMMA=";S:PRINT
100 PRINT "IL NUMERO DEGLI ELEMENTI E'";N
110 END
```

Programma 6

8.6 La rappresentazione di numeri

Precedentemente nel corso, abbiamo lasciato aperti un mucchio di problemi nella rappresentazione dei numeri nei nostri programmi. Non intendiamo affrontare la matematica della rappresentazione dei numeri nei computer in generale, ma abbiamo bisogno di riordinarci le idee sui numeri.
In termini generali, i computer devono essere capaci di trattare ed immagazzinare i seguenti tipi di numeri:
(a) numeri interi positivi o negativi
(b) frazioni o numeri che hanno una parte intera e una parte frazionaria (numeri di misura)
(c) numeri molto grandi o molto piccoli
(d) il numero zero

L'unico consiglio fondamentale che possiamo darti a questo stadio è che se un numero è stato "implicato" in un qualche tipo di calcolo all'interno di un programma, allora devi considerarlo con un po' di cautela. La ragione di questa affermazione è che i numeri all'interno di un computer sono immagazzinati e manipolati in forma binaria e cioè in base 2, piuttosto che nella base 10 che ben conosciamo (numerazione decimale).
Nella notazione decimale a noi familiare ricorderai che un numero come 1/3 o 1/7, non può avere una rappresentazione esatta, per es. 1/3 = 0.3333... Assumiamo che, aggiungendo tutte le cifre 3 che vogliamo a questo numero, possiamo raggiungere un livello accettabile di precisione in un problema particolare. Nel sistema binario, allo stesso modo, alcuni numeri non possono essere espressi con esattezza. Per es. in forma decimale possiamo dire che 1/10=0.1 esattamente, ma quando questo numero viene cambiato in forma binaria, non può essere rappresentato con esattezza.

La maggior parte degli interpreti BASIC permette di esprimere i numeri con una precisione di sei cifre decimali. In forma decimale, quindi, il numero 3, potrebbe essere rappresentato come 3.10000 con una precisione di 6 cifre decimali. Comunque, se questo numero è il risultato di un qualche calcolo all'interno del computer, l'output del computer potrebbe essere 3.09999 o 3.10001. Perciò, in generale, devi diffidare sempre dell'ultima cifra significativa in una risposta (e cioé la cifra all'estrema destra del numero).

Lo abbiamo visto nell'output del programma 4.

Il programma 7 mostra come può avvenire questo tipo di inesattezza. Il ciclo FOR...NEXT... aggiunge 4.0, 4.1 e 4.25 nelle locazioni S, T e U un centinaio di volte.
Ora 4.0 e 4.25 sono rappresentati con precisione in forma binaria, mentre 4.1 non lo è. Quindi, il risultato di somme ripetute è preciso per ciò che concerne 4.0 e 4.25; non lo è per 4.1 dove l'errore è 0.04 in 4100.00. Naturalmente, eviteremo di scrivere programmi che richiedano tali ripetizioni, laddove sia possibile, ma speriamo che il programma evidenzi le possibili inesattezze.

```
10 REM **DIMOSTRAZIONE DI PRECISIONE**
15 PRINT CHR$(147)
20 LET S=0
30 LET T=0
40 LET U=0
50 FOR I=1 TO 1000
60 LET S=S+4.0
```

```
70 LET T=T+4.1
80 LET U=U+4.25
90 NEXT I
100 PRINT S,T,U
110 END
```

Programma 7

```
RUN
 4000          4100.00016          4250

READY
```

- ESEGUI IL PROGRAMMA 7 -

Numeri grandi e piccoli

Sei cifre decimali non ci permettono di lavorare con numeri molto piccoli o molto grandi, i quali, pertanto, sono rappresentati in BASIC in forma esponenziale.

Numeri piccoli

0.000586321 significa $\frac{586321}{1,000,000,000} = \frac{586321}{10^9}$

che potremmo esprimere anche come 586321 x 10^{-9}.
Potremmo inoltre esprimere 0.000586321 come

$$\frac{0.586321}{1000} = 0.586321 \times 10^{-3}$$

In BASIC, quest'ultimo numero sarebbe scritto come 0.56321E—3 o 0.586321E—03 ed in modo simile,
0.0234539 = 2.34539 x 10^{-2} = 2.34539E—2

e

0.00000000959734 = 0.959734 x 10^{-2}8 =
0.959734E—8

E sta per esponente e la base per l'elevazione a potenza è 10. Perciò E-4 significa muovere il punto decimale di 4 posizioni a sinistra, e E+9 significa muovere il punto decimale di 9 posizioni a destra.

Numeri grandi

12368500 = 1.23685 x 10^7 = 1.23685E+7
935.432 = 0.935432 x 10^3 = 0.935432E+3
959734000000000000000 = 0.959734E + 21

La maggior parte degli interpreti BASIC hanno un intervallo di variazione che va almeno da E—32 a E+32; l'intervallo di variazione del Commodore 64 va da 2.93873588E—39 a 1.70141183E+38

```
10 REM **RAPPRESENTAZIONE DEI NUMERI**
15 PRINT CHR$(147)
20 PRINT"NUMERO","RAPPRESENTAZIONE"
30 FOR I=-10 TO 10
40 PRINT"10^";I,10^I
50 NEXT I
60 END
```

Programma 8

- ESEGUI IL PROGRAMMA 8 -

```
RUN
NUMERO     RAPPRESENTAZIONE
10^-10      9.99999998E-11
10^-9       1E-09
10^-8       1E-08
10^-7       1E-07
10^-6       1E-06
10^-5       1E-05
10^-4       1E-04
10^-3       1E-03
10^-2       .01
10^-1       .1
10^ 0       1
10^ 1       10
10^ 2       100
```

```
10^ 3       1000
10^ 4       10000
10^ 5       100000
10^ 6       1000000
10^ 7       10000000
10^ 8       100000000
10^ 9       1E+09
10^ 10      1E+10

READY
```

8.7 La funzione INT di arrotondamento

In alcuni problemi, abbiamo bisogno di arrotondare il risultato dell'operazione al più vicino numero intero.

Per es.: nel caso di 6.6 il numero intero più vicino è 7 e nel caso di 7.4 il più vicino numero intero è ugualmente 7.
Questo è vero specialmente se non siamo sicuri dell'esattezza dell'ultima cifra di un numero decimale,

es. 6.99999 o 7.00001 sta per 7.

La funzione INT(X+0.5) fa questo arrotondamento per noi ed il seguente programma lo dimostra.

```
10 REM **INT PER L'ARROTONDAMENTO**
15 PRINT CHR$(147)
20 PRINT" X","INT(X)","INT(X+0.5)":PRINT
30 FOR I=-1.4 TO -2.6 STEP(-.1)
40 PRINT I,INT(I),INT(I+0.5)
50 NEXT I
60 END
```

```
RUN
 X          INT(X)     INT(X+0.5)

-1.4        -2         -1
-1.5        -2         -1
-1.6        -2         -2
-1.7        -2         -2
-1.8        -2         -2
-1.9        -2         -2
-2          -2         -2
-2.1        -3         -2
-2.2        -3         -2
-2.3        -3         -2
-2.4        -3         -2
-2.5        -3         -2
-2.6        -3         -3

READY
```

Programma 9a

Cambiando la linea 30 in 30 FOR I = 1.4 TO 2.6 STEP(.1) otterremo:

```
RUN
 X          INT(X)     INT(X+0.5)

 1.4         1          1
 1.5         1          2
 1.6         1          2
 1.7         1          2
 1.8         1          2
 1.9         1          2
 2           2          2
 2.1         2          2
 2.2         2          2
 2.3         2          2
 2.4         2          2
 2.5         2          2
 2.6         2          3

READY
```

Programma 9b

- ESEGUI IL PROGRAMMA 9a e 9b -

DAV3

Che cosa verrà fuori se cambiamo la linea 30 del precedente programma in:

(a) 30 FOR I=4 TO 2.2 STEP (.2)?
(b) 30 FOR I=4 TO –6 STEP (–.2)?

8.8 La funzione ABS

Un'altra funzione aritmetica interessante è quella che trova il "modulo" o il valore assoluto di un numero. Sembra una cosa piuttosto difficile, ma è molto semplice.

ABS(X) ci dà semplicemente il valore positivo di X.
es. ABS(23)=23, ABS(–23)=23

Il programma seguente illustra la funzione ABS:

```
10 REM **LA FUNZIONE ABS**
15 PRINT CHR$(147)
20 PRINT"{DESTRA}X"," Y","X+Y","ABS(X+Y)":PRINT
30 FOR I=1 TO 4
40 READ X,Y
50 PRINT X,Y,X+Y,ABS(X+Y)
60 NEXT I
70 END
100 DATA 5,7,5,-7,-5,7,-5,-7
```

Programma 10

```
RUN
 X          Y          X+Y        ABS(X+Y)

 5          7           12         12
 5         -7          -2          2
-5          7           2          2
-5         -7          -12         12

READY
```

- ESEGUI IL PROGRAMMA 10 -

DAV4

Cosa verrà fuori se cambiamo la linea 100 in
100 DATA 9,14,11,–2,–2,13,–7,–8?

8.9 Iterazione

Il processo di formulare un'ipotesi su un valore, controllarlo, formulare una ipotesi migliore e controllarlo una seconda volta, ecc., è detto iterazione.

L'essenza del concetto di iterazione è costituita da:

- prendere un punto di partenza arbitrario
- ipotizzare l'accuratezza del risultato che si vuole ottenere
- precisarlo con una sequenza di operazioni ripetute.

Calcolo della radice quadrata per iterazione.

Il BASIC può calcolare direttamente le radici quadrate, come dimostra questo programma:

```
5 REM **RADICI QUADRATE CALCOLATE DIRETTAMENTE**
7 PRINT CHR$(147)
10 FOR X=33 TO 63 STEP(10)
20 PRINT X;TAB(5);X^(.5);SQR(X)
30 NEXT X
40 END
```

Programma 11

(SQR(X) dà la radice quadrata di X, dato che X> =0.)

```
RUN
 33    5.74456265  5.74456265
 43    6.55743853  6.55743853
 53    7.28010989  7.28010989
 63    7.93725393  7.93725393

READY
```

Ti illustriamo come puoi trovare una radice quadrata con un metodo iterati-

vo, non perché tu debba farlo normalmente, ma perché è uno degli esempi più semplici per dimostrare il concetto di iterazione.

Il metodo

Se vuoi calcolare la radice quadrata di un numero N:

- fai un'ipotesi sulla radice quadrata, diciamo G
- esegui N/G
- quindi la media tra G e N/G è un'ipotesi migliore di quella che hai formulato in precedenza, e cioè è più vicina, rispetto alla prima ipotesi, alla radice quadrata sconosciuta
- ricomincia usando questa nuova ipotesi "migliore".

Non faremo qui la prova (tutti i libri di testo di matematica la riportano) ma mostreremo come lavora in un caso molto semplice.

Calcola la radice	$N=12$
Ipotesi	$G=2$
Operazione	$N/G=6$
Fai la media	$\frac{G+N/G}{2}=\frac{2+6}{2}=4$

Perché 4 è la nuova ipotesi:

Ipotesi	$G=4$
	$N/G=3$
	$\frac{G+N/G}{2}=3.5$

3.5 é quindi la nuova ipotesi

In forma di tabella, tutto questo diventa:

G	N/G	$\frac{G+N/G}{2}$
2	12/2=6	(2+6)/2=4
4	12/4=3	(4+3)/2=3.5
3.5	12/3.5=. . .	. . . .

DAV5

Per essere sicuro di aver colto il significato di questo processo, disegna una tabella simile alla precedente ma poni a 1 la prima ipotesi.

Iterazione...stop

La domanda che ora ci poniamo è la seguente:
"come facciamo a fermare questo processo?"
Beh, fermeremo il processo quando la radice quadrata della nostra ipotesi è più vicina possibile a N. Cosa vogliamo dire con "più vicina possibile?" La risposta è lasciata a te. È una grossa responsabilità. Quanto vuoi che sia accurata la radice quadrata?
Se, per es., vuoi trovare la radice quadrata di 12 fino alla seconda posizione decimale, allora la differenza tra G*G e N dovrebbe essere

<0.005

Non abbiamo bisogno di sapere se G*G sia maggiore o minore di N. Ci interessa solo la differenza. Torniamo quindi ad ABS. Se

ABS(N—G*G)<0.005

allora fermati: é quello che faremo anche noi, qui di seguito.

Algoritmo descrittivo del calcolo della radice quadrata per iterazione

1. Inizio.
2. Immetti il numero di cui vuoi calcolare la radice quadrata.
3. Immetti il livello di precisione che pretendi.
4. Inserisci un'ipotesi di radice quadrata.
5. Se l'ipotesi corrisponde al livello di precisione richiesto, vai a 8, altrimenti continua con l'istruzione successiva.
6. Fai un'ipotesi pií precisa.
7. Torna a 5
8. Visualizza il valore trovato della radice quadrata.
9. Stop.

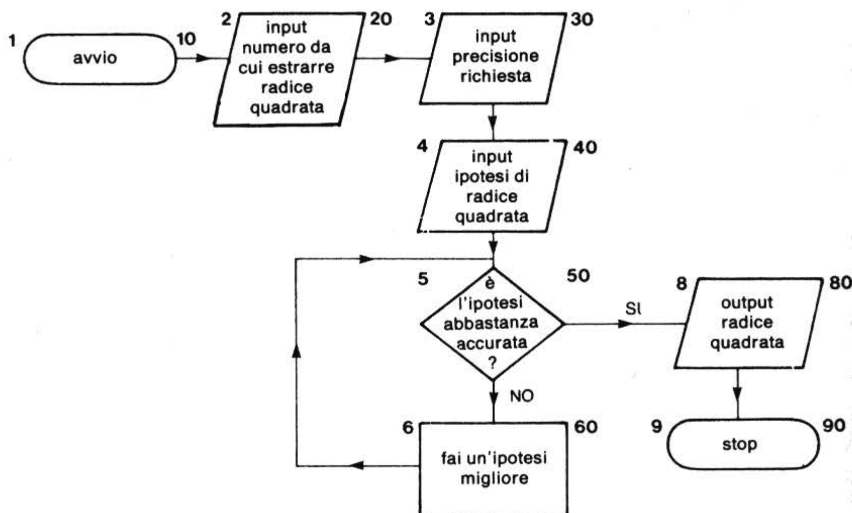


Figura 8 – Diagramma di flusso per il calcolo della radice quadrata per iterazione.

```
10 REM **RADICE QUADRATA PER ITERAZIONE**
15 PRINT CHR$(147)
20 INPUT"NUMERO";N
30 INPUT"PRECISIONE RICHIESTA";A
40 INPUT"IPOTESI";G
45 REM **INIZIO DELL'ITERAZIONE**
50 IF ABS(N-G*G)<A THEN 80
60 LET G=(G+(N/G))/2
70 GOTO 45
75 REM **OUTPUT**
80 PRINT
81 PRINT"LA RADICE QUADRATA DI";N;"E'";G
90 END
```

Programma 12 - Iterazione per la radice quadrata

```
RUN
NUMERO? 12
PRECISIONE RICHIESTA? .005
IPOTESI? 2

LA RADICE QUADRATA DI 12 E' 3.46428572

READY
```

```
RUN
NUMERO? 12
PRECISIONE RICHIESTA? .00005
IPOTESI? 2

LA RADICE QUADRATA DI 12 E' 3.46410162

READY
```

- ESEGUI IL PROGRAMMA 12 -

Tracing del processo iterativo

Il risultato del programma 12 non é molto sorprendente. Come abbiamo sottolineato prima, possiamo trovare N direttamente, con il computer. Ma volevamo un esempio semplice per dimostrare come lavora l'iterazione, perciò guarderemo con più attenzione cosa è successo e faremo una traccia del programma 12:

```
41 PRINT
43 PRINT" VECCHIA"," NUOVA"," ABS(N-G*G)":PRINT
55 PRINT G,;
65 PRINT G,ABS(N-G*G)
```

Ogni passaggio (e cioé ogni iterazione) attraverso le linee 55 e 65 del ciclo stamperà un resoconto di come è avvenuto il calcolo.

```
10 REM **RADICE QUADRATA PER ITERAZIONE**
15 PRINT CHR$(147)
20 INPUT"NUMERO";N
30 INPUT"PRECISIONE RICHIESTA";A
40 INPUT"IPOTESI";G
41 PRINT
43 PRINT" VECCHIA"," NUOVA","  ABS(N-G*G)":PRINT
45 REM **INIZIO DELL'ITERAZIONE**
50 IF ABS(N-G*G)<A THEN 80
55 PRINT G;TAB(10);
60 LET G=(G+N/G)/2
65 PRINT G;TAB(22);ABS(N-G*G)
70 GOTO 45
75 REM **OUTPUT**
```

```
80 PRINT
81 PRINT"LA RADICE QUADRATA DI";N;"E'";G
90 END
```

Programma 13 - Radice quadrata per iterazione con tracing

```
RUN
NUMERO? 12
PRECISIONE RICHIESTA? .005
IPOTESI? 2

 VECCHIA    NUOVA        ABS(N-G*G)

 2          4            4
 4          3.5          .25
 3.5        3.46428572   1.27551204E-03

LA RADICE QUADRATA DI 12 E' 3.46428572

READY
```

solo 3 cicli

```
RUN
NUMERO? -4
PRECISIONE RICHIESTA? .0005
IPOTESI? 2

 VECCHIA    NUOVA        ABS(N-G*G)

 2          0            4
 0
?DIVISION BY ZERO  ERROR IN 60
READY
```

per aumentare la precisione di dieci volte, sono necessari solo 4 cicli.

A seconda della grandezza del numero, se immettiamo un numero negativo, il Commodore 64 dare un messaggio di errore (come ha fatto sopra) o dare numeri e caratteri incomprensibili. In tal caso, dovrai far terminare il programma con RUN STOP.

```
RUN
NUMERO? 12
PRECISIONE RICHIESTA? .00005
IPOTESI? 2

 VECCHIA    NUOVA        ABS(N-G*G)

 2          4            4
 4          3.5          .25
 3.5        3.46428572   1.27551204E-03
 3.46428572   3.46410162   3.38040991E-08

LA RADICE QUADRATA DI 12 E' 3.46410162

READY
```

- ESEGUI IL PROGRAMMA 13 -

```
RUN
NUMERO? -4 ———————————————— nota il numero
PRECISIONE RICHIESTA? .005       negativo
IPOTESI? 2
```

Esercizio 9

Cambia il precedente programma di calcolo della radice quadrata in un programma che calcoli la radice cubica. Se G é un'ipotesi di radice cubica di N, allora $1/2(G+N/G^2)$ costituirà un'ipotesi migliore.

Compito 8

1. Scrivi una tabella di conversione per convertire i litri in galloni e viceversa, e falla stampare sullo schermo. Arrotonda le risposte in litri al numero intero pií vicino e quelle in galloni alla seconda cifra decimale, usando il seguente rapporto: un gallone equivale 4.544 litri.

Note: Metti i numeri da 1 a 15 nella parte inferiore dello schermo, stampa gli equivalenti in litri in basso a destra; cioé la prima riga dovrebbe avere 5 sotto la colonna dei litri, 1 alla colonna centrale e 0.22 nella colonna dei galloni, perché un gallone equivale a 5 litri e un litro equivale a 0.22 galloni.

2. Scrivi un programma per calcolare e stampare il consumo di benzina della tua macchina per la distanza percorsa dall'ultimo pieno di serbatoio e la quantità di benzina necessaria a riempire il serbatoio.

Suggerimento: dovresti usare la funzione INT(X+0.5) per ottenere livelli di precisione ragionevoli.

Se te la senti, potresti adattare il programma del consumo di combustibile anche per il rapporto miglia/gallone e Km/litro! (1 miglio = 1.609 Km; 1 gallone = 4.544 litri).

Obiettivi dell'unitê 8

Calcolare (manualmente) una media aritmetica ☐

Scrivere un programma per calcolare medie aritmetiche ☐

Scrivere un programma per trovare il più grande e il più piccolo elemento in una lista di dati ☐

Usare *, / e ↑ nei programmi ☐

Eseguire il dry running di un programma ☐

Interpretare i numeri nella notazione E ☐

Usare INT(X+.5) per arrotondare ☐

Usare ABS(X) ☐

Scrivere programmi con routines iterative che includano procedure terminali ☐

Inserire linee di stampa del tracing in un programma ☐

Risposte DAV e esercizi

DAV1

Somma = 8+4+2+6+1+7+6+1+4 = 39
Ci sono 9 numeri.
Cosè la media aritmetica é 39/9 = 4.333 ...

Esercizio 1

```
10 REM **LUNGHEZZA MEDIA**
15 PRINT CHR$(147)
17 PRINT"CALCOLO LUNGHEZZA MEDIA"
18 PRINT
20 LET S=0
30 LET C=1
35 REM **INIZIO DEL CICLO DI LETTURA**
40 READ W$
50 IF W$="ZZZZ" THEN 150
60 LET L=LEN(W$)
70 LET S=S+L
80 LET C=C+1
90 GOTO 35
100 DATA LA,PIGRIZIA,ANDO',AL,MERCATO,ED
    ,UN,CAVOLO,COMPRO'
110 DATA MEZZOGIORNO,ERA,SUONATO,QUANDO,
    A,CASA,RITORNO'
120 DATA PRESE,L',ACQUA,ACCESE,IL,FUOCO,
    SI,SEDETTE,E,RIPOSO'
130 DATA ED,INTANTO,A,POCO,A,POCO,ANCHE
140 DATA IL,SOLE,TRAMONTO',ZZZZ
150 REM *************
160 LET N=C-1
170 LET A=S/N
180 PRINT" LA LUNGHEZZA MEDIA DELLE PARO
    LE E' DI"
185 PRINT A;"CARATTERI"
190 END
```

Programma 14

```
RUN
CALCOLO LUNGHEZZA MEDIA

 LA LUNGHEZZA MEDIA DELLE PAROLE E' DI
 4.5 CARATTERI

READY
```

- ESEGUI IL PROGRAMMA 14 -

Esercizio 2

```
10 REM **MEDIA DI 100 LANCI**
20 REM **DI UN DADO**
25 PRINT CHR$(147)
30 PRINT"LANCIO DI UN DADO PER 100 VOLTE"
40 PRINT
50 LET S=0
60 FOR I=1 TO 100
70 LET X=INT(6*RND(1)+1)
80 LET S=S+X
90 NEXT I
100 PRINT"PUNTEGGIO MEDIO=";S/100
110 END
```

Programma 15

- ESEGUI IL PROGRAMMA 15 -

Esercizio 3

```
10 REM **MEDIA DI 100 LANCI**
20 REM **DI DUE DADI**
25 PRINT CHR$(147)
30 PRINT"LANCIO DI DUE DADI PER 100 VOLTE"
40 PRINT
50 LET S=0
60 FOR I=1 TO 100
70 LET X=INT(6*RND(1)+1)
80 LET S=S+X+Y
85 LET Y=INT(6*RND(1)+1)
90 NEXT I
100 PRINT"PUNTEGGIO MEDIO=";S/100
110 END
```

Programma 16

- ESEGUI IL PROGRAMMA 16 -

Esercizio 4

```
10 REM **ISTOGRAMMA**
15 PRINT CHR$(147)
20 DIM S(100)
30 FOR K=0 TO 100 STEP(10):S(K)=0:NEXT K
40 REM *************
50 READ M
60 IF M=-9999 THEN 140
70 REM *************
80 K=10
85 REM **TROVA L'INTERVALLO DI VARIAZIONE**
90 IF M<K THEN 115
100 K=K+10
110 GOTO 85
115 REM **MEMORIZZARE I DATI**
120 S(K-10)=S(K-10)+1
130 GOTO 40
140 REM *************
150 FOR K=0 TO 100 STEP(10)
160 PRINT K,S(K)
170 NEXT K
180 REM *************
190 DATA 56,47,52,65,24,34,59,37,49,66
200 DATA 38,24,62,76,31,47,66,61,74,45
210 DATA 66,44,55,67,36,56,54,54,50,43
220 DATA 18,83,23,79,29,-9999
```

(riga 120: trovata la categoria corretta, incrementa il relativo contatore)

Programma 17

```
RUN
 0           0
 10          1
 20          4
 30          5
 40          6
 50          8
 60          7
 70          3
 80          1
 90          0
 100         0

READY
```

- ESEGUI IL PROGRAMMA 17 -

Esercizio 5

```
10 REM **SOMMA DI RECIPROCHI**
15 PRINT CHR$(147)
20 LET S=0
30 LET N=1
35 REM **INIZIO DEL CICLO DI CALCOLO**
40 LET S=S+1/N
50 IF S>2.4 THEN 80
60 LET N=N+1
70 GOTO 35
80 REM **OUTPUT**
90 PRINT"SOMMA=";S
95 PRINT"IL NUMERO DEGLI ELEMENTI E'";N
100 END
```

Programma 18

```
RUN
SOMMA= 2.45
IL NUMERO DEGLI ELEMENTI E' 6

READY
```

- ESEGUI IL PROGRAMMA 18 -

Esercizio 6

```
10 REM **SOMMA DI N^(-2)**
15 PRINT CHR$(147)
20 S=0
30 N=1
35 REM **INIZIO DEL CICLO DI CALCOLO**
40 S=S+N^(-2)
50 IF S>1.5 THEN 80
60 N=N+1
70 GOTO 35
80 REM **OUTPUT**
90 PRINT"SOMMA=";S
95 PRINT"IL NUMERO DEGLI ELEMENTI E'";N
100 END
```

Programma 19

```
RUN
SOMMA= 1.51179705
IL NUMERO DEGLI ELEMENTI E' 7

READY
```

Puoi, naturalmente, cambiare la linea 50 per stabilire il numero di elementi necessari perché la somma superi altri valori, per es.:

```
50 IF S>1.6 THEN 80
```

```
RUN
SOMMA = 1.60049693
IL NUMERO DEGLI ELEMENTI E' 22←per 1.6
READY
```

```
50 IF S>1.61 THEN 80
```

```
RUN
SOMMA = 1.61103901
IL NUMERO DEGLI ELEMENTI E' 29 ← per 1.61
READY
```

```
50 IF S>1.62 THEN 80
```

```
RUN
SOMMA  = 1.62024396
IL NUMERO DEGLI ELEMENTI E' 40← per 1.62
READY
```

```
50 IF S>1.63 THEN 80
```

```
RUN
SOMMA = 1.63011952
IL NUMERO DEGLI ELEMENTI E' 67← per 1.63
READY
```

Un buon punto di partenza per questi e simili programmi potrebbe essere un ciclo di lezioni sui limiti e la convergenza di serie di numeri.

- ESEGUI IL PROGRAMMA 19 -

Esercizio 7

```
10 REM **FATTORIALI**
15 PRINT CHR$(147)
20 PRINT"PER FINIRE SCRIVI -9999"
30 PRINT
35 REM **INPUT SUCCESSIVO**
40 INPUT"FATTORIALE SUCCESSIVO";N
50 IF N=-9999 THEN 125
60 LET F=1
70 FOR I=1 TO N
80 LET F=F*I
90 NEXT I
100 PRINT N,F
110 PRINT
120 GOTO 35
125 REM **FATTO**
130 END
```

Programma 20

```
RUN
PER FINIRE SCRIVI -9999

FATTORIALE SUCCESSIVO? 1
 1         1

FATTORIALE SUCCESSIVO? 3
 3         6

FATTORIALE SUCCESSIVO? 5
 5         120

FATTORIALE SUCCESSIVO? 7
 7         5040

FATTORIALE SUCCESSIVO? 9
 9         362880

FATTORIALE SUCCESSIVO? 11
 11        39916800

FATTORIALE SUCCESSIVO? -9999

READY
```

- ESEGUI IL PROGRAMMA 20 -

DAV2

PASSO N.	LINEA N.	N	S
1	20	0	0
2	30	1	0
3	40	1	1
4	60	2	1
5	40	2	5
6	60	3	5
7	40	3	14
8	60	4	14
9	40	4	30
10	60	5	30
11	40	5	55

Esercizio 8

```
10 REM **INTERESSE COMPOSTO**
12 PRINT CHR$(147)
15 PRINT"INTERSSE COMPOSTO"
17 PRINT
20 PRINT"IMMETTI DEPOSITO,PERCENTALE,TEMPO"
25 PRINT
30 INPUT D,P,T
35 PRINT
40 PRINT"TEMPO","RICAVO({LIRE})"
45 PRINT
50 FOR I=1 TO T
60 PRINT I,D*(1+P/100)^I
70 NEXT I
80 END
```

Programma 21

```
RUN
INTERESSE COMPOSTO

IMMETTI DEPOSITO,PERCENTUALE,TEMPO

? 500,11.5,5
```

```
TEMPO       RICAVO(\)

 1           557.5
 2           621.612501
 3           692.097938
 4           772.804201
 5           861.676685

READY
```

- ESEGUI IL PROGRAMMA 21 -

DAV3

(a)

X	INT(X)	INT(X+0.5)
.4	0	0
.6	0	1
.8	0	1
1	1	1
1.2	1	1
1.4	1	1
1.6	1	2
1.8	1	2
2	2	2

(b)

X	INT(X)	INT(X+0.5)
.2	0	0
0	0	0
–.2	–1	0
–.4	–1	0
–.6	–1	–1

DAV4

RUN

X	Y	Y+Y	ABS(X+Y)
9	14	23	23
11	–2	9	9
–2	13	11	11
–7	–8	–15	15

DAV5

9	N/G	$\frac{G+N/G}{2}$
1	12	6.5
6.5	1.8	4.15
4.15	2.89	3.52
3.52	3.41	3.46 ecc.

Esercizio 9

```
10 REM **RADICE CUBICA E ITERAZIONE**
15 PRINT CHR$(147)
20 INPUT"NUMERO";N
30 INPUT"PRECISIONE RICHIESTA";A:PRINT
40 LET G=N/2
43 LET C=1
45 REM **INIZIO DEL CALCOLO**
50 IF ABS(N-G*G*G)<A THEN 75
60 LET G=0.5*(G+N/(G*G))
67 LET C=C+1
70 GOTO 45
75 REM **OUTPUT**
80 PRINT"IL NUMERO DI CICLI =";C:PRINT
81 PRINT"LA RADICE CUBICA DI";N;"=";G
90 END
```

Programma 22

```
RUN
NUMERO? 28
PRECISIONE RICHIESTA? .005

IL NUMERO DI CICLI = 14

LA RADICE CUBICA DI 28 = 3.03640957

READY


RUN
NUMERO? 10101
PRECISIONE RICHIESTA? .005
```

```
IL NUMERO DI CICLI = 28

LA RADICE CUBICA DI 10101 = 21.6166341

READY


RUN
NUMERO? -937
PRECISIONE RICHIESTA? .005

IL NUMERO DI CICLI = 21

LA RADICE C   CA DI-937 =-9.78541983

READY
```

- ESEGUI IL PROGRAMMA 22 -

UNITÀ 9
Introduzione al trattamento dei dati

9.1 Introduzione

In questa unità, ritorneremo di nuovo sull'importanza di stabilire un qualche ordinamento all'interno dei dati su cui lavoriamo. In particolare, analizzeremo dettagliatamente un metodo di ordinamento dei dati: quello della procedura di interscambio. Una volta ordinati i dati, ti mostreremo come trovarli velocemente usando la procedura di ricerca per bisezione. In ultimo, vedremo come manipolare i dati in forma tabellare.
Queste attività ci permettono di capire come le subroutines ci aiutino ad eseguire molti di quei piccoli compiti ripetitivi che si possono trovare in un programma di qualsiasi dimensione.

9.2 Ordinamento

Nell'unità didattica 4 abbiamo parlato molto della procedura per trovare

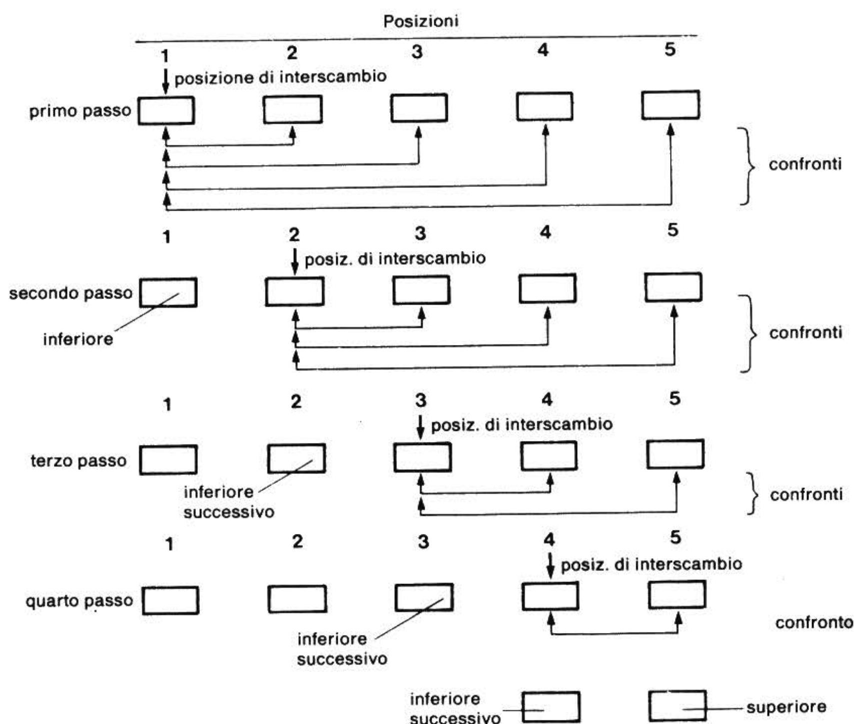


Figura 1 – Procedura di ordinamento di una lista di 5 elementi.

l'elemento di valore inferiore in una lista. Una procedura di interscambio metteva l'elemento più basso in posizione 1 e ripeteva la stessa operazione per il resto della lista, mettendo il valore più basso successivo in posizione 2, etc. Ricorderai che ti abbiamo assegnato il compito di riordinare l'intera lista. Data l'importanza dell'ordinamento per interscambio, ora lo analizzeremo più dettagliatamente.

La figura 1 illustra la procedura per mettere gli elementi nelle locazioni da 1 a 5: il più basso in 1, il successivo in 2 e così via.

Primo stadio: tutti gli elementi vengono confrontati con quello che si trova in posizione 1 e se uno di questi è inferiore, allora viene posto in posizione 1.

Secondo stadio: la posizione 1 può essere ignorata e la procedura ripetuta sulle posizioni da 2 a 5. Il valore più basso successivo verrà trovato e posto in posizione 2.

Terzo stadio: le posizioni 1 e 2 vengono ignorate. La procedura è ripetuta sulle posizioni da 3 a 5.

Quarto stadio: vengono considerati soltanto gli elementi 4 e 5. Quello di valore inferiore andrà in quarta posizione, mentre quello di valore superiore andrà direttamente in quinta posizione e non saranno necessarie altre fasi.

Possiamo riassumere questa procedura di ordinamento come:

Numero di ciclo	Punto di interscambio	sottosequenza rimanente	
		inizio	fine
1	1	2	5
2	2	3	5
3	3	4	5
4	4	5	5

Figura 2a – Quattro ordinamenti in una lista di 5 elementi.

O, più in generale, se vogliamo ordinare una lista di N elementi:

numero di ciclo	punto di interscambio	sottosequenza rimanente	
		Inizio	Fine
1	1	2	N
2	2	3	.
3	3	4	.
.	.	.	.
.	.	.	N
.	.	.	.
K	K	K+1	.
.	.	.	.
.	.	.	N
.	.	.	N
N−1	N−1	N	N

Figura 2b – (N–1) ordinamenti in una lista di N elementi.

Poiché per ogni fase viene effettuata una serie ripetitiva di confronti, ovviamente ricorreremo al ciclo FOR...NEXT... E poi ci servirà un ciclo ulteriore per stabilire quante volte deve girare il ciclo stesso.

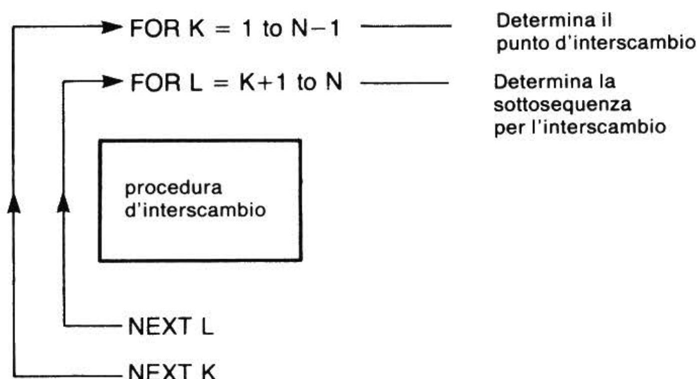


Figura 3 – I cicli nidificati dell'ordinamento per interscambio.

DAV1

Ordina per interscambio i numeri che seguono ed indica come vengono immagazzinati dopo ogni esecuzione;

6, 1, 4, 0, 2, 3, 7, 8

Il programma è:

se l'elemento nella subsequenza > = all' elemento nella posizione di interscambio allora non deve essere scambiato

il ciclo più esterno decide il punto di interscambio

```
210 REM **ROUTINE D'ORDINAMENTO**
220 FOR K=1 TO N-1
230 FOR L=K+1 TO N
240 IF X$(L)>=X$(K) THEN 275
250 T$=X$(L)
260 X$(L)=X$(K)
270 X$(K)=T$
275 REM **SALTA QUI SE LA LISTA E' ORDINATA**
280 NEXT L
290 NEXT K
300 REM **FINE DELL'ORDINAMENTO**
```

si potrebbero condensare in un'unica linea per accelerare l'esecuzione

il ciclo interno decide la sotto–sequenza

Programma 1 – Ordinamento per interscambio.

Uso del programma di ordinamento

Ti proponiamo qui un uso particolare del programma di ordinamento per organizzare una lista di nomi in ordine alfabetico.
le linee 50–80 leggono i dati
le linee 210–300 eseguono l'ordinamento
le linee 410–450 stampano la lista ordinata
I dati sono stati immagazzinati alla linea 900

```
10 REM **ROUTINE DI ORDINAMENTO**
15 PRINT CHR$(147)
20 PRINT"{DESTRA}{DESTRA}{DESTRA}{DESTRA}{DESTRA}
{DESTRA}...ROUTINE DI ORDINAMENTO..."
```

```
25 PRINT:PRINT
30 DIM X$(100)
50 I=1
55 REM **CICLO DI LETTURA**
60 READ X$
70 IF X$="ZZZZ" THEN 180
80 X$(I)=X$:I=I+1:GOTO 55
180 REM ******
190 N=I-1:REM **LUNGHEZZA LISTA**
200 REM ******
210 REM **ROUTINE DI ORDINAMENTO**
220 FOR K=1 TO N-1
230 FOR L=K+1 TO N
240 IF X$(L)>=X$(K) THEN 280
250 T$=X$(L)
260 X$(L)=X$(K)
270 X$(K)=T$
280 NEXT L
290 NEXT K
300 REM **FINE DELLA ROUTINE
    DI ORDINAMENTO**
400 REM ******
410 PRINT"LISTA FINALE ORDINATA"
415 PRINT
420 FOR P=1 TO N
430 PRINT X$(P);""
440 NEXT P
450 PRINT
500 REM ******
900 DATA TOMMASO,SANDRO,PIETRO,GIACOMO,SERGIO
,ZZZZ

READY.
```

routine di ordinamento (righe 210–300)

stampa dei risultati (righe 410–450)

Programma 2 – Uso della routine di ordinamento

```
RUN
      ...ROUTINE DI ORDINAMENTO...


LISTA FINALE ORDINATA

GIACOMO
PIETRO
SANDRO
SERGIO
TOMMASO


READY
```

9.3 SUBROUTINE

A questo punto, avrai capito che i programmi hanno una struttura generale e che sono composti di parti più piccole come i paragrafi di un testo. Si usa suddividere il programma nelle parti significative che lo costituiscono e scrivere e controllare ogni parte separatamente. Spesso, in un programma, alcune operazioni vengono ripetute diverse volte. Tali operazioni sono dette subroutine e contribuiscono a semplificare e a chiarire la struttura di un programma.
Illustreremo il funzionamento delle subroutine dando un'occhiata finale alla procedura di ordinamento. Inseriremo due ulteriori linee per la stampa del tracing nel programma. In tal modo, vedremo cosa accade in ognuno dei tre stadi della routine di ordinamento:

Routine di ordinam.	**Linea di stampa del tracing**
1. Input	La lista, così come viene immessa
2. Ordinamento	La lista dopo ogni sotto–sequenza
3. Output	La lista finale, riordinata

La figura 5 illustra la struttura complessiva e l'uso di una subroutine PRINT per tutte le tre operazioni PRINT.

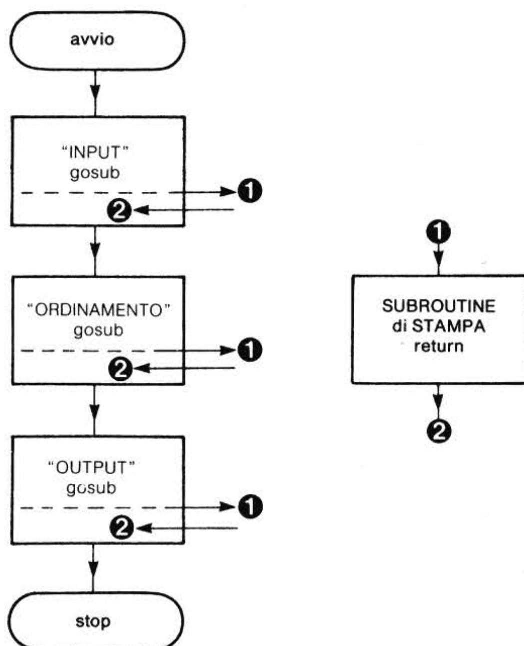


Figura 5 – Sub–routine di stampa nel programma di ordinamento.

GOSUB

Nel BASIC, per andare ad una subroutine diciamo:

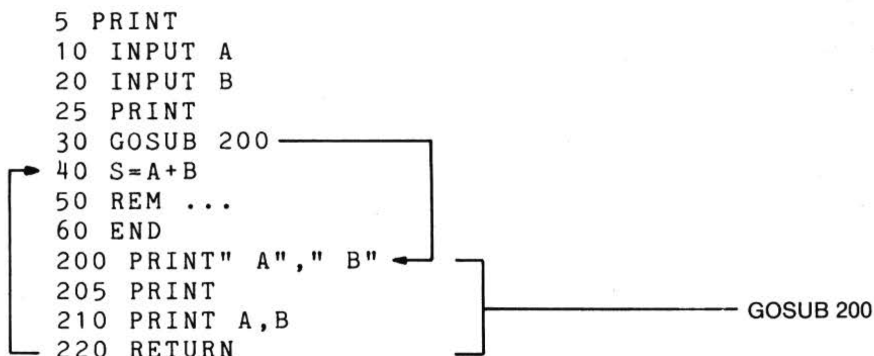
GOSUB

seguito dal numero di linea dell'inizio della subroutine. Ogni subroutine deve finire con l'istruzione

RETURN

che farà tornare il controllo alla linea successiva all'istruzione GOSUB nel corpo principale del programma. Perciò, nel seguente segmento di programma:
la linea 30 trasferisce il controllo alla linea 200,
vengono eseguite le linee 200 e 210,

la 220 riporta il controllo alla linea 40,
il programma continua in modo normale.



DAV2

Qual'è il valore di **B** dopo l'esecuzione del programma:
(a) se viene immesso 5?
(b) se viene immesso 3?

```
5 REM **DAV2**
7 PRINT CHR$(147)
10 INPUT A
15 PRINT
20 IF A<5 THEN 35
30 GOSUB 65
35 REM **SALTA QUI**
40 B=A*A
50 PRINT" ";B
60 END
65 REM **SUBROUTINE**
70 Q=1/A
80 RETURN
```

Programma 3

Quello che segue è un programma di ordinamento con una subroutine di stampa (linee 500–550), usata ogni volta che vengono eseguite le linee 194, 280 e 420.

```
10 REM **ROUTINE DI ORDINAMENTO**
15 PRINT CHR$(147)
20 PRINT"{DESTRA}{DESTRA}{DESTRA}{DESTRA}{DESTRA}
{DESTRA}...ROUTINE DI ORDINAMENTO..."
30 DIM X$(100)
50 I=1
55 REM **CICLO DI LETTURA**
60 READ X$
70 IF X$="ZZZZ" THEN 180
80 X$(I)=X$:I=I+1:GOTO 60
180 REM ************
190 N=I-1:REM **LUNGEZZA DELLA LISTA**
192 PRINT"LISTA ALL'INIZIO"
194 GOSUB 510 ┐
196 PRINT     ┘──────────── Prima traccia
200 REM ************
210 REM **ROUTINE DI ORDINAMENTO**
220 FOR K=1 TO N-1
225 PRINT"PASSAGGIO NO.";K
230 FOR L=K+1 TO N
240 IF X$(L)>X$(K) THEN 275
250 T$=X$(L):X$(L)=X$(K):X$(K)=T$
275 REM **SALTA QUI SE LA LISTA E' IN ORDINE*
*
280 GOSUB 510 ┐
285 NEXT L    ┘──────────── Seconda traccia
287 PRINT
290 NEXT K
300 REM **FINE DELLA ROUTINE DI ORDINAMENTO**

400 REM ************
410 PRINT"LISTA FINALE ORDINATA"
420 GOSUB 510 ┐
450 GOTO 450  ┘──────────── Output
500 REM **SUBROUTINE DI STAMPA**  ┐
510 FOR P=1 TO N                  │
520 PRINT X$(P);" ";              │
530 NEXT P                        ├── Subroutine
540 PRINT                         │
550 RETURN                        ┘
900 DATA TOMMASO,SANDRO,PIETRO,GIANNI,SERGIO,
ZZZZ
READY.
```

Programma 4 - Subroutine di stampa in un programma di ordinamento

```
RUN
        ...ROUTINE DI ORDINAMENTO...
LISTA ALL'INIZIO
TOMMASO SANDRO PIETRO GIANNI SERGIO

PASSAGGIO NO. 1
SANDRO TOMMASO PIETRO GIANNI SERGIO
PIETRO TOMMASO SANDRO GIANNI SERGIO
GIANNI TOMMASO SANDRO PIETRO SERGIO
GIANNI TOMMASO SANDRO PIETRO SERGIO

PASSAGGIO NO. 2
GIANNI SANDRO TOMMASO PIETRO SERGIO
GIANNI PIETRO TOMMASO SANDRO SERGIO
GIANNI PIETRO TOMMASO SANDRO SERGIO

PASSAGGIO NO. 3
GIANNI PIETRO SANDRO SERGIO TOMMASO

LISTA FINALE ORDINATA
GIANNI PIETRO SANDRO SERGIO TOMMASO
```

Stampato da GOSUB alla linea 194

ogni blocco viene stampato da GOSUB alla linea 280 nelle quattro occasioni in cui il programma esegue il ciclo controllato da K

stampata da GOSUB alla linea 420

Esempi con subroutine

Lo scopo di una subroutine è quello di semplificare e abbreviare programmi lunghi. Per sua natura, quindi, è difficile ottenere brevi programmi significativi che illustrino le subroutines senza incorrere in una qualche artificiosità. Abbiamo bisogno di un programma in cui la stessa funzione o una simile venga ripetuta in punti differenti.

Esempio 1

Il gioco dei dadi può servirci da esempio. Un paio di dadi viene lanciato due volte e il punteggio totale di ogni lancio viene annotato a parte. Se i due punteggi totali sono gli stessi, il gioco finisce; se sono diversi, i dadi vengono lanciati un'altra volta. Scrivi un programma che simuli il gioco e stampi il numero di lanci necessari ad ottenere due punteggi uguali ed il valore del punteggio.

Soluzione

```
10 REM **LANCI EGUALI**
20 PRINT CHR$(147)
25 PRINT"SIMULAZIONE DEL GIOCO DEI DADI":PRINT
30 C=1
40 GOSUB 130:REM **PRIMO LANCIO**
50 S1=S
60 GOSUB 130:REM **SECONDO LANCIO**
70 S2=S
80 IF S1=S2 THEN 100
90 C=C+1:GOTO 40
100 PRINT"PUNTEGGIO UGUALE";S1;"IN";C;"LANCI"
110 END
120 REM **SUBROUTINE DI LANCIO**
130 D1=INT(6*RND(1)+1):D2=INT(6*RND(1)+1)
140 S=D1+D2:RETURN
```

la subroutine produce valori differenti di S nel programma principale

subroutine

Programma 5 – Simulazione del gioco di azzardo.

```
RUN
SIMULAZIONE DEL GIOCO DEI DADI

PUNTEGGIO UGUALE 8 IN 3 LANCI

READY


RUN
SIMULAZIONE DEL GIOCO DEI DADI

PUNTEGGIO UGUALE 7 IN 3 LANCI

READY


RUN
SIMULAZIONE DEL GIOCO DEI DADI

PUNTEGGIO UGUALE 4 IN 4 LANCI

READY
```

```
RUN
SIMULAZIONE DEL GIOCO DEI DADI

PUNTEGGIO UGUALE 10 IN 7 LANCI

READY
```

Esercizio 1

(a) Scrivi un segmento di programma per stampare una linea di 40 trattini sullo schermo
(b) Scrivi una linea di programma per stampare un "sottomarino" con i segni < = >, in una posizione (dello schermo o della stampante) determinata dalla variabile S.
(c) Scrivi un programma per stampare su linee successive:
   (i) una riga di trattini;
   (ii) un sottomarino in un punto definito;
   (iii) un'altra riga di trattini,
   in modo che una subroutine stampi le linee da (i) a (iii).

Esercizio 2

Perché non usiamo anche per il sottomarino una subroutine? Invece di battaglie navali in un mare bidimensionale, abbiamo un sottomarino in un canale monodimensionale. In ogni modo, col nostro disegno potremo fare un gioco molto semplice.
Scrivi un programma per generare un numero casuale tra 1 e 36. Usa il numero casuale per stampare il sottomarino in posizioni a caso nel canale.

Esercizio 3

L'essenza del gioco che stiamo per fare con la macchina sarà chiara dall'esercizio 2. Il computer genera un numero casuale e ti invita a trovare il sottomarino scegliendo un numero tra 1 e 36. Se hai indovinato la posizione esatta, e cioè tra S e S+2, se S è il numero casuale (ricorda che il sottomarino occupa tre posizioni su una linea), allora la macchina registra "colpita" e il gioco finisce. Se non trovi il sottomarino, la macchina registrerà "mancato" e ti inviterà a riprovare.

Scrivi un programma per fare ciò.
Attento! Devi scrivere un programma che ti dia la possibilità di fermare il gioco prima di trovare il sottomarino, perché è irritante dover cercare tutte le posizioni sullo schermo solo per fermare il programma. Dovresti staccare la spina della corrente ed allora sarebbe proprio frustrante!.

9.4 Ricerca

Il problema del sottomarino ci introduce molto bene alla questione della ricerca di dati. Il solo sistema metodico di trovare un sottomarino era quello di cercare nel canale le posizioni una dopo l'altra partendo da una delle due estremità. Quanto più facile sarebbe disporre di un programma che dica "vicino" o "lontano", in modo appropriato, dopo ogni tentativo.

Senza dubbio puoi pensare immediatamente ad una procedura per trovare il sottomarino nel modo più rapido possibile!
In modo simile, se i dizionari, gli elenchi telefonici, le enciclopedie, i cataloghi delle biblioteche non fossero organizzati in ordine alfabetico, pensa a quanto sarebbe difficile trovare l'informazione desiderata!

Ma se abbiamo avuto un sacco di problemi nell'ordinare i nostri dati alfabeticamente o numericamente, ora avremmo bisogno di una tecnica di ricerca efficace per trovare un qualsiasi elemento dato. Se consultiamo un vocabolario o un elenco telefonico per cercare un nome, non dobbiamo cominciare con lo sfogliare la prima pagina e percorrere il volume metodicamente, pagina per pagina finché non troviamo il nostro nome.

Facciamo piuttosto un tentativo molto rozzo, ad esempio, se il nome comincia con "P" proveremo ad aprire l'elenco più o meno verso la metà e cominceremo a cercare da quel punto.

La ricerca per bisezione

"Un rozzo tentativo" è un'espressione troppo imprecisa per un computer.
Comunque possiamo specificare i tentativi come segue:

- dividi il numero degli elementi a metà e chiedi: l'elemento si trova sopra o sotto la metà

- se si trova sotto, allora definiamo un nuovo campo con l'elemento di mezzo che ora fa da limite superiore
- se si trova sopra, allora l'elemento di mezzo è il nostro limite inferiore
- in entrambi i casi dividiamo a metà il numero degli elementi precedenti e ripetiamo la nostra procedura di bisezione con il nuovo insieme di elementi.

Facciamo un esempio di ricerca per bisezione:
il numero 7 si trova nella lista 1, 2, 3, 4, 5, 6, 7, 8, 9, 10?

Lista ordinata | 1 2 3 4 5 6 7 8 9 10 |

Dividi a metà la lista | 1 2 3 4 5 | 6 7 8 9 10 |

7 è l'elemento di mezzo? No
7 è < dell'elemento di mezzo? No
Perciò 7 si trova nella metà superiore

Dividi la lista a metà | 6 7 8 9 10 |

7 è l'elemento di mezzo? No
7 è < dell'elemento di mezzo? Si

Dividi la lista a metà | 6 7 8 |

7 è l'elemento di mezzo? Si
Allora 7 si trova nella lista.

Questo breve schema illustra i principi della ricerca per bisezione ma in pratica abbiamo bisogno di distinguere tra gli elementi in una lista e gli indici di quegli elementi.

Esempio 2

Una lista ordinata contiene gli elementi A, F, I, M, P, T, U, Z. Usa la procedura di ricerca per bisezione per scoprire se P si trova o no nella lista.

Chiamiamo P, la nostra "Richiesta", il valore che vogliamo cercare:

Indice	1	2	3	4	5	6	7	8
Elemento	A	F	I	M	P	T	U	Z
Inizio–Indice	INF(1)			⋮				SUP(8)
Indice di mezzo, Int $\frac{(1+8)}{2}=4$				METÀ(4)				

Confronti

> la richiesta = elemento (4)? no!
> la richiesta < elemento (4)? no!
> l'indice (4) diventa l'inferiore

Indice	4	5	6	7	8
Elemento	M	P	T	U	Z
Inizio–Indice inferiore	INF(4)		⋮		SUP(8)
Indice di mezzo, Int $\frac{(4+8)}{2} = 6$			METÀ(6)		

Confronti

> la richiesta = elemento (6)? no!
> la richiesta < elemento (6)? si!
> l'indice (6) diventa il superiore

Indice	4	5	6
Elemento	M	P	T
Inizio–Indice inferiore	INF(4)	⋮	SUP(6)
Indice di mezzo, Int $\frac{(4+6)}{2} = 5$		METÀ(5)	

Confronti

> la richiesta = elemento (5)? si!

Figura 6 – Ricerca per bisezione.

Esercizio 4

Applica la procedura di ricerca per bisezione alla lista dell'esempio 2, ma per cercare la lettera I.

Dobbiamo fare solo tre confronti per trovare l'elemento "P" nell'esempio 2. L'intera procedura sembrerebbe essere più vantaggiosa della semplice ricerca su tutta la lista, ma l'efficacia del metodo non appare su liste brevi. Dimostreremo più avanti la sua efficienza nella ricerca su liste più lunghe, ma prima dobbiamo concludere il discorso fin qui fatto, con qualche chiarimento.

Alcuni problemi nella ricerca per bisezione

(a) come fermarsi

Esempio 3

Esegui la stessa procedura precedente, ma cerca la lettera "Q".

Il metodo sarebbe esattamente lo stesso fino al terzo confronto, perciò cominciamo da qui.

	4	5	6
Elemento	M	P	T
Inizio–Indice inferiore	INF(4)	:	SUP(4)
Indice di mezzo, Int $\frac{(4+6)}{2} = 5$		METÀ(5)	

Confronti

la richiesta = elemento (5)? no!
la richiesta < elemento (5)? no!
l'indice (5) diventa l'inferiore

Indice	4	5	6
Elemento		P	T
Inizio–Indice inferiore		INF(5)	SUP(6)
Indice di mezzo, Int $\frac{(5+6)}{2} = 5$		METÀ(5)	

Confronti

la richiesta= elemento (5)? no!
la richiesta<...ma, non lo abbiamo già fatto?

A questo punto, sembra proprio che non siamo capaci di fermarci. Q non c'è ma ci siamo bloccati a cercarlo tra P e T. Nell'ultimo esempio abbiamo incontrato il problema di bloccare il processo. Se gli indici "inferiore" e "superiore" sono diventati così vicini da trovarsi in posizioni adiacenti e la "richiesta" non è stata trovata, allora non è un membro della lista. Siamo arrivati alla fine ed è questo il risultato della ricerca. Finiamo, cioè, o quando troviamo la "richiesta" o quando gli indici superiore e inferiore sono adiacenti (Superiore – Inferiore = 1).

(b) Come cominciare

Avviare il processo sembrava una cosa abbastanza facile. Facciamo sì che Indice (1) = inferiore e Indice (N) = superiore. Si incorrerebbe in qualche problema se l'elemento(1) e l'elemento (N) non fossero effettivamente l'inferiore e il superiore.
Consideriamo ad esempio la lista seguente, che non contiene lettere prima di C o dopo di S.

1	2	3	4	5
C	F	G	P	S
inferiore				superiore

Se la richiesta fosse A o B o una lettera superiore ad S, allora il processo non avrebbe luogo. La soluzione più facile è quella di assicurarsi che gli elementi alle estremità della lista abbiano sempre valori estremi, es. in una lista di nomi, AAAA è l'elemento (1) e ZZZZ è l'elemento (N).

Tracceremo ora l'algoritmo in forma di diagramma di flusso.

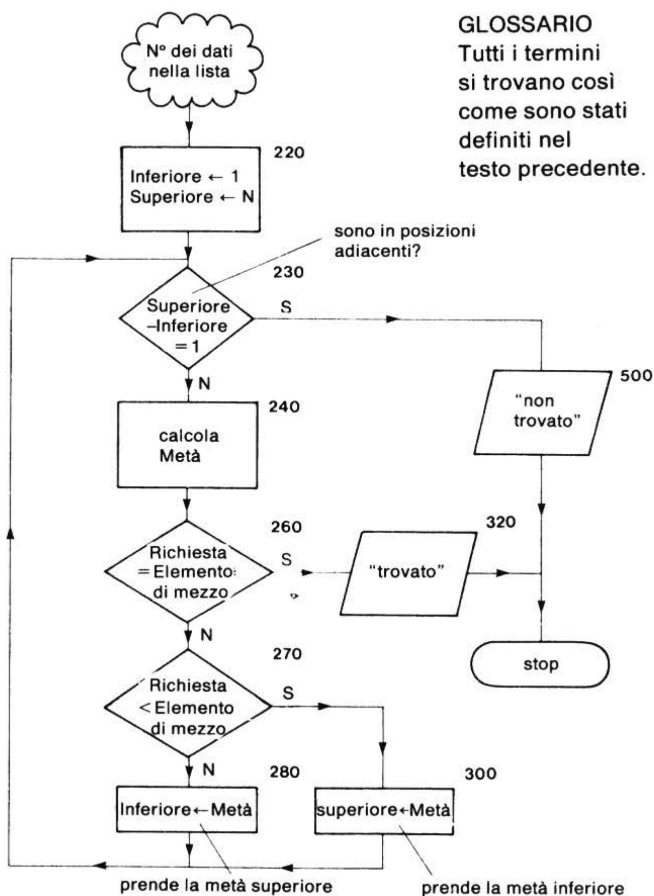


Figura 7 – Diagramma di flusso della ricerca per bisezione.

Tutto quello che dobbiamo fare è scrivere il programma.

```
10 REM **RICERCHE PER BISEZIONE**
20 PRINT CHR$(147)
30 DIM N$(20):DIM T$(20)
40 I=1
45 REM **INIZIO DEL CICLO DI LETTURA**
50 READ N$(I),T$(I)
60 IF N$(I)="ZZZZ" THEN 90
70 I=I+1:GOTO 45
90 REM ******
100 N=I:REM **ORA USIAMO ZZZZ**
110 REM ******
150 INPUT"NOME RICHIESTO";Q$
160 PRINT
200 REM **INIZIO DELLA RICERCA**
210 PRINT"INF";TAB(5);"SUP";TAB(11);"M";TAB(15);
"N$(M)"
215 PRINT
220 L=1:H=N
230 M=INT((L+H)/2)
240 IF M=L THEN 490
250 PRINT L;TAB(4);H;TAB(10);M;TAB(15);N$(M)
255 PRINT
260 IF Q$=N$(M) THEN 320
270 IF Q$<N$(M) THEN 300
280 L=M:GOTO 230
300 H=M:GOTO 230
320 REM **FINE DELLA RICERCA**
330 PRINT"IL NUMERO TELEFONICO DI ";Q$;" E' ";
T$(M)
340 PRINT
350 GOTO 600
490 REM **IL NOME NON C'E'**
500 PRINT Q$;" NON E' NELLA LISTA"
510 PRINT
600 INPUT"VUOI CERCARE UN ALTRO NOME";R$
605 PRINT
610 IF R$="SI" THEN 110
620 END
900 DATA AAAA,0000
910 DATA ALFREDO,1234
920 DATA ANDREA,9823
```

Le linee 210 e 250 impostano il tabulato

```
930 DATA CONSUELO,1850
940 DATA DANIELE,7294
950 DATA LAURA,5821
960 DATA MARCO,8632
970 DATA MAURO,7832
980 DATA MORENO,1383
990 DATA SERGIO,1147
1000 DATA STEFANO,5529
1010 DATA UMBERTO,9936
1020 DATA ZZZZ,9999
```

Programma 6 – Programma di ricerca per bisezione

```
RUN
NOME RICHIESTO? MAURO

INF  SUP   M   N$(M)

 1   13    7   MARCO

 7   13    10  SERGIO

 7   10    8   MAURO

IL NUMERO TELEFONICO DI MAURO E' 7832

VUOI CERCARE UN ALTRO NOME? SI

NOME RICHIESTO? STEFANIA

INF  SUP   M   N$(M)

 1   13    7   MARCO

 7   13    10  SERGIO

 10  13    11  STEFANO

STEFANIA NON E' NELLA LISTA

VUOI CERCARE UN ALTRO NOME? NO


READY
```

9.5 Tabelle

Quando vogliamo immagazzinare molte informazioni possiamo usare vari metodi. Uno è quello di metterle in liste (vedi unità 4) chiamate anche vettori. Un secondo metodo è creare una tabella o matrice.

Supponiamo che tu voglia immagazzinare i seguenti dati:

	1° TRIM	2° TRIM	3° TRIM	4° TRIM
Vendita	20	70	80	40
Servizio	10	14	18	11
Carburante	30	45	50	30

Figura 8 – Guadagni di una stazione di servizio (x 1.000.000)

Potresti scrivere tutti i dati in una lista ma sarebbe difficile utilizzarla. I primi 4 elementi rappresentano il guadagno per la vendita delle macchine, i successivi 4 il guadagno per il servizio, etc.
Altrimenti, dovresti avere 3 liste: una per le macchine vendute, una per il servizio e una per il carburante. Ma il BASIC ti permette di avere una tabella bidimensionale chiamata con uno dei 286 nomi di variabili per es.:

T(,)

Confronto di liste e tabelle

Le liste hanno bisogno di un indice che indichi la posizione dell'elemento nella lista. Le tabelle necessitano di 2 indici che, generalmente, sono sottoscritti.

Lista

L(1),L(2),L(3)...L(I)...

└ indice di questo elemento = 3

Matrice

A(1,1)	A(1,2)	A(1,3)
A(2,1)	A(2,2)	A(2,3)
A(3,1)	A(3,2)	A(3,3)

└ questo elemento necessita di 2 indici:
3 ci dice che si trova sulla riga 3;
2 ci dice che si trova sulla colonna 2.

Tabelle

- Una tabella deve contenere o solo variabili–stringhe o solo variabili numeriche. (I numeri possono naturalmente essere immagazzinati come stringhe e il loro valore trovato dalla funzione VAL).
- Usiamo uno dei 286 nomi di variabile per descrivere le tabelle come un insieme per es. tabelle B$, tabella M3$.

Generalmente, una tabella include:

	col.1	col.2	col.3	col.4
riga 1	r1c1	r1c2	r1c3	....
riga 2	r2c1	r2c2	r2c3	....
riga 3	r3c1	r3c2	r3c3	....
ecc.	....	....	....	....

Figura 9 – Righe e colonne di una tabella.

Per i dati della stazione di servizio, T necessita di 3 righe e 4 colonne, e contiene, perciò, 12 elementi:

T(1,1)	T(1,2)	T(1,3)	T(1,4)
T(2,1)	T(2,2)	T(2,3)	T(2,4)
T(3,1)	T(3,2)	T(3,3)	T(3,4)

Quindi

T(2,1) = 10
T(3,3) = 50 ecc.

Tutto questo è molto simile alle tabelle che hai incontrato precedentemente. Abbiamo detto che un file consiste di una serie di registrazioni ognuna delle quali consiste, a sua volta, di campi. In forma di tabella dovrebbe apparire come:

	Campo 1	Campo 2
	Nome	Numero di telefono
Record 1	ALFREDO	1234
Record 2	ANDREA	9823
Record 3	CONSUELO	1850
Record 4	DANIELE	7294

o più genericamente:

	Campo 1	Campo 2	Campo 3	ecc.
Record 1	R1F1	R1F2	R1F3	....
Record 2	R2F1	R2F2	R2F3	....
Record 3	R3F1	R3F2	R3F3	....
ecc.	....	....	....	....

se la tabella dei numeri di telefono è detta T$ allora gli elementi individuali saranno etichettati:

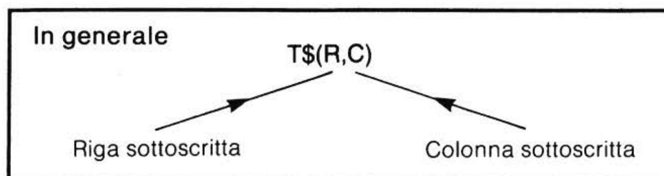
	Campo 1	Campo 2
	Nome	Numero di telefono
Record 1	T$(1,1)=Alfredo	T$(1,2)=1234
Record 2	T$(2,1)= Andrea	T$(2,2)=9823
Record 3	T$(3,1)= Consuelo	T$(3,2)=1950
Record 4	T$(4,1)= Daniele	T$(4,2)=7294

- l'intera tabella è chiamata tabella T$.
- Ogni elemento nella tabella è descritto da due indici sottoscritti. Dunque 1950 (3° fila, 2° colonna) è:

  T$(3,2)

- Il 3 e il 2 descrivono la posizione dell'elemento T$(3,2); non il suo valore Il suo valore è 1950.
  Perciò, diciamo:

  T$(3,2) = 1950



Esempio 4

La tabella N$ ha 9 valori. Quali sono i nomi della variabile?

	R \ C	1	2	3
N$(,)	1	ALFREDO	ANDREA	CONSUELO
	2	DANIELE	LAURA	MARCO
	3	MAURO	MORENO	SERGIO

Soluzione

ALFREDO = N$(1,1) ANDREA = N$(1,2) CONSUELO = N$(1,3)
DANIELE = N$(2,1) LAURA = N$(2,2) MARCO = N$(2,3)
MAURO = N$(3,1) MORENO = N$(3,2) SERGIO = N$(3,3)

DAV3

Nella seguente tabella A$ identifica le variabili e i loro valori, come nell'es. 4.

ALFREDO	ANDREA	CONSUELO
DANIELE	FRANCESCO	LAURA
MARCO	MAURO	MORENO
NICOLETTA	SANDRO	SERGIO
STEFANO	UMBERTO	VALENTINA

Tabelle e cicli nidificati

Se i cicli FOR...NEXT e le liste sembrano essere fatti l'uno per l'altro, così anche i cicli FOR...NEXT... e le tabelle sembrano complementari.
Per esempio, supponiamo che tu voglia leggere:

ALFREDO, ANDREA, CONSUELO, DANIELE, FRANCESCA, LAURA, MARCO, MAURO, MORENO, NICOLETTA, SANDRO, SERGIO, STEFANO, UMBERTO, VALENTINA

in una tabella N$, con 4 righe e 4 colonne (abbiamo bisogno di una matrice–stringa perché stiamo immagazzinando dati sotto forma di stringhe). Possiamo usare l'istruzione READ in 2 cicli nidificati:

```
60 FOR I=1 TO 4
90 FOR J=1 TO 4

80 READ N$(I,J)
90 NEXT J
100 NEXT I
```

Il processo è eseguito completamente dalle linee da 10 a 100 del Programma 7. In questo modo immagazziniamo il valore in una tabella, ma non possiamo vedere il risultato finché non lo stampiamo. La seconda metà del programma stampa la tabella di valori in una colonna con I e J associandoli in modo da identificare chiaramente come sono usati I e J.

```
10 REM **LEGGE E STAMPA UNA TABELLA**
20 PRINT CHR$(147)
30 DIM N$(20,5) ———————————— Il comando DIM
40 REM **ROUTINE DI LETTURA**     forma una matrice,
60 FOR I=1 TO 4                    in questo caso, di
70 FOR J=1 TO 4                    20 righe per 5 colonne
80 READ N$(I,J)
90 NEXT J
100 NEXT I
110 REM **ROUTINE DI STAMPA**
115 PRINT" I"," J","N$(I,J)"
120 PRINT
130 FOR I=1 TO 4 ———————————————— Righe
140 FOR J=1 TO 4 ———————————————— colonne
150 PRINT I,J,N$(I,J)
160 NEXT J
170 NEXT I
190 GOTO 270
240 REM **********
250 DATA ALFREDO,ANDREA,CONSUELO,DANIELE,FRANCESCO
260 DATA LAURA,MARCO,MAURO,MORENO,NICOLETTA,SANDRO
270 DATA SERGIO, STEFANO,UMBERTO,VALENTINA,VITTORIO
280 END
```

Programma 7 – Lettura dei dati in ordine 4 x 4.

```
RUN
 I            J           N$(I,J)

 1            1           ALFREDO
 1            2           ANDREA
 1            3           CONSUELO
 1            4           DANIELE
 2            1           FRANCESCO
 2            2           LAURA
 2            3           MARCO
 2            4           MAURO
 3            1           MORENO
 3            2           NICOLETTA
 3            3           SANDRO
 3            4           SERGIO
 4            1           STEFANO
 4            2           UMBERTO
 4            3           VALENTINA
 4            4           VITTORIO

READY
```

DAV4

Sul programma 7 sono state fatte delle correzioni che riportiamo qui sotto. Come risulterà la tabella finale?

```
60 FOR I=1 TO 3
70 FOR J=1 TO 5
130 FOR I=1 TO 3
140 FOR J=1 TO 5
```

Tabella di output

L'output del programma 7 non ci permette di vedere i dati della matrice in forma di tabella. Cancelliamo la linea 115 e inseriamo una nuova routine di stampa:

```
130 FOR I=1 TO 5
140 FOR J=1 TO 3
150 PRINT TAB(10*(J-1));N$(I,J);
160 NEXT J
```

la prima colonna comincia alla posizione 1–1 = 0

colonne ampie 10 caratteri

```
170 PRINT
180 NEXT I
190 GOTO 270
```

Programma 8

Quindi l'output è:

```
ALFREDO      ANDREA       CONSUELO
DANIELE      FRANCESCO    LAURA
MARCO        MAURO        MORENO
NICOLETTA    SANDRO       SERGIO
STEFANO      UMBERTO      VALENTINA
```

Compito 9

1. Un venditore ha 4 linee di prodotti. I valori (in lire) degli ordini della sua azienda in una settimana sono illustrati nella tabella:

giorno \ prodotto	1	2	3	4	totali
1	500	300	20	25	e
2	600	700	40	0	f
3	200	550	60	20	g
4	250	450	100	5	h
5	400	200	100	11	i
totali	a	b	c	d	t

Scrivi un programma che lo aiuti ad analizzare il lavoro della settimana dando:

(i) i totali giornalieri (e, f, g, h, i)
(ii) i totali dei prodotti (a, b, c, d)
(iii) il totale settimanale complessivo (t).

2. Scrivi un programma per ingrandire il gioco del sottomarino ad una griglia 10x10. Se l'ipotesi è vicina al sottomarino allora il programma darà il segnale "mancato per poco".

Obiettivi dell'unità 9

Ora che hai completato questa unità, controlla se sei in grado di:

Usare l'ordinamento per interscambio (manualmente su un insieme di dati)	☐
Scrivere due cicli di programma nidificati per eseguire un ordinamento per interscambio.	☐
Seguire GOSUB nei programmi	☐
Scrivere GOSUB nei programmi	☐
Usare la ricerca per bisezione (manualmente) su un insieme di dati	☐
Scrivere un programma di ricerca per bisezione	☐
Mettere dati in matrici	☐
Scrivere un programma per leggere dati in una matrice	☐
Scrivere un programma per stampare dati da una matrice	☐
Scrivere un programma per trovare la somma delle righe e delle colonne in una matrice	☐

Risposte alle DAV e agli esercizi

DAV1

0	6	4	1	2	3	7	8
0	1	6	4	2	3	7	8
0	1	2	6	4	3	7	8
0	1	2	3	6	4	7	8
0	1	2	3	4	6	7	8
0	1	2	3	4	6	7	8
0	1	2	3	4	6	7	8

Lista finale ordinata

0 1 2 3 4 6 7 8

DAV2

(a) B = 1/25 (che si scrive 4E–2 – e cioè 0.004)
(b) B = 9 (GOSUB non è usato in questo caso)

Esercizio 1

(a)
```
10 FOR I=1 TO 40
20 PRINT"-";
30 NEXT I
40 PRINT
```

Programma 9

o, in una linea:

FOR I = 1 TO 40: PRINT "–";:NEXT I:PRINT

(b) PRINT TAB(B);"< = >"

(c)
```
10 INPUT S
20 GOSUB 100
30 PRINT TAB(S);"<=>"
40 GOSUB 100
50 END
100 FOR I=1 TO 39:PRINT"-";:NEXT I:PRINT
110 RETURN
```

Programma 10

Il valore S che diamo determinerà la posizione del sottomarino lungo il canale. La figura che avremo sarà:

```
RUN
? 4
---------------------------------------
    <=>
---------------------------------------

READY
```

Poiché il Commodore 64 avanza automaticamente di una riga, quando si arriva alla fine del disegno ci fermeremo alla posizione 39 invece che alla posizione 40.

Esercizio 2

```
10 REM **SOTTOMARINO**
20 PRINT CHR$(147)
50 S=INT(36*RND(1)+1)
60 GOSUB 510
70 PRINT TAB(S);"<=>"
80 GOSUB 510
```

```
90 END
500 REM **SUBROUTINE DI STAMPA**
510 FOR I=1 TO 39:PRINT"-";:NEXT I:PRINT
520 RETURN
```

Programma 11

```
RUN
---------------------------------------
                          <=>
---------------------------------------

READY


RUN
---------------------------------------
                <=>
---------------------------------------

READY
```

Esercizio 3

```
10 REM **SOTTOMARINO**
20 PRINT CHR$(147)
30 REM **STAMPA BATTAGLIA**
40 GOSUB 300                                        →1
50 PRINT"CON UN NUMERO TRA 1 E 36 MI PUOI SCOVARE"
60 GOSUB 300                                        →2
70 REM **POSIZIONE CAUSUALE DEL SOTTOMARINO**
80 S=INT(36*RND(1)+1)
90 PRINT
100 INPUT"PROVA UN ALTRO NUMERO";X
110 IF X<S THEN 190
120 IF X>S+2 THEN 190
130 REM **COLPITO**
140 GOSUB 300                                       →3
150 PRINT TAB(S);"COLPITO"
160 GOSUB 300                                       →4
170 GOTO 320
180 REN **ERRORE**
190 GOSUB 300                                       →5
200 PRINT"SBAGLIATO"
```

```
210 GOSUB 300                                    →6
220 INPUT"VUOI GIOCARE ANCORA";R$
230 IF R$="SI" THEN 80
240 PRINT"CIAO! ERO QUI!"
250 GOSUB 300                                    →7
260 PRINT TAB(S);"<=>"
270 GOSUB 300                                    →8
280 GOTO 320
290 REM **SUBROUTINE DI STAMPA**
300 FOR I=1 TO 39:PRINT"-";:NEXT I:PRINT
310 RETURN
320 END
```

la subroutine, usata 8 volte, stampa una riga di trattini

Programma 12

```
RUN
---------------------------------------
CON UN NUMERO TRA 1 E 36 MI PUOI SCOVARE
---------------------------------------

PROVARE UN ALTRO NUMERO? 27
---------------------------------------
SBAGLIATO
---------------------------------------
VUOI GIOCARE ANCORA? SI

PROVA UN ALTRO NUMERO? 30
---------------------------------------
SBAGLIATO
---------------------------------------
VUOI GIOCARE ANCORA? NO
CIAO! ERO QUI!
---------------------------------------
     <=>
---------------------------------------

READY
```

Esercizio 4

1	2	3	4	5	6	7	8
A	F	I	M	P	T	U	Z

Indice (4) di mezzo

Indice di mezzo = INT $\frac{(1+8)}{2} = 2$

> Richiesta = elemento(4)? No.
> Richiesta < elemento(4)? Si.
> Fa sì che l'indice (4) sia l'indice superiore

A	F	I	M
1	2	3	4

Indice (2) di mezzo

Indice di mezzo = INT $\frac{(1+4)}{2} = 2$

> Richiesta = elemento(2)? No.
> Richiesta < elemento(2)? No.
> Fa sì che l'indice(2) sia l'indice inferiore

2	3	4
F	I	M

Indice di mezzo

Indice di mezzo = INT $\frac{(2+4)}{2} = 3$

> Richiesta = elemento(3)? Si.
> Perciò I è nella lista

DAV 3

```
A$(1,1)=ALFREDO     A$(1,2)=ANDREA     A$(1,3)=CONSUELO
A$(2,1)=DANIELE     A$(2,2)=FRANCESCO  A$(2,3)=LAURA
A$(3,1)=MARCO       A$(3,2)=MAURO      A$(3,3)=MORENO
A$(4,1)=NICOLETTA   A$(4,2)=SANDRO     A$(5,3)=SERGIO
A$(5,1)=STEFANO     A$(5,2)=UMBERTO    A$(5,3)=VALENTINA
```

DAV4

```
RUN
 I          J          N$(I,J)

 1          1          ALFREDO
 1          2          ANDREA
 1          3          CONSUELO
 1          4          DANIELE
 1          5          FRANCESCO
 2          1          LAURA
 2          2          MARCO
 2          3          MAURO
 2          4          MORENO
 2          5          NICOLETTA
 3          1          SANDRO
 3          2          SERGIO
 3          3          STEFANO
 3          4          UMBERTO
 3          5          VALENTINA

READY
```

(Nota che VITTORIO non è stato letto nella tabella. Una tabella 5x3 leggerà soltanto i primi 15 elementi).

UNITÀ 10
Manipolazione di file

10.1 Conservare programmi

Fino ad ora il corso si è occupato di un sistema con tre soli dispositivi e cioè una tastiera, l'elaboratore e il monitor. Ma tale sistema è "volatile", e cioè, quando lo spegni, tutti i tuoi sforzi di programmazione vengono perduti. Se vuoi eseguire di nuovo un programma, dovrai ribatterlo. Ma tutti i microcomputer permettono di conservare un programma su un comune nastro di audiocassetta. Una volta che il programma è sulla cassetta, puoi spegnere il calcolatore ma puoi far rieseguire ancora il programma quando vuoi, nel futuro.

In due occasioni ti sarà utile memorizzare i programmi:
(a) quando vuoi conservare un programma completo;
(b) quando vuoi conservare parte di un programma che stai sviluppando. Se conservi il tuo lavoro ogni volta che hai lo schermo pieno, e se capita un qualche accidente alla parte su cui staï lavorando, non perdi mai molto. Quando poi conservi il programma completo, puoi cancellare le parti intermedie.

Il processo di registrazione

Come prima cosa devi dare al tuo programma un nome, che il computer userà per conservarlo. I sistemi prevedono lunghezza e caratteri diversi, per il nome del file. Il Commodore 64 permette di usare fino a 16 caratteri.

Supponiamo che voglia conservare un programma chiamato NOMEPROG. La procedura sarà:

- Collegare al computer il registratore di cassette
- Battere SAVE "NOMEPROG" sul computer e premere il tasto RETURN
- Avviare il registratore (premere i tasti play e record)
- Lo schermo della televisione diventerà vuoto (azzurrino)
- Una volta che il programma è caricato, spegnere il registratore.

Il processo di caricamento

Una volta memorizzato il programma, lo potrai riutilizzare.
Lo puoi ricaricare nella macchina con un comando LOAD.
Un'interazione tipica sarebbe:

- Collegare al tuo computer il registratore Commodore 1530(C2N)
- Battere LOAD "NOMEPROG" e premere il tasto RETURN
- Accendere il registratore
- Spegnere il registratore quando il caricamento ha avuto luogo

Oppure

- Collegare il registratore al computer
- Battere LOAD" " e premere RETURN
- Accendere il registratore
- Spegnere il registratore quando il caricamento ha avuto luogo.

10.2 Stampare con una stampante

Se vuoi il listing di un programma sulla stampante, devi avvertire la stampante che quell'informazione è in arrivo.
Usa il comando OPEN: "OPEN numero di file logico, numero di dispositivo".
Per la stampante VIC–151, il numero di dispositivo è 4 o 5, il numero di file logico può essere un numero qualsiasi tra 0 e 255. Puoi cambiare il numero per adattarlo alle tue esigenze, ricordandoti di utilizzarlo per tutto l'insieme di comandi.

OPEN 1,4
lo farà.

L'altra cosa di cui abbiamo bisogno è trasferire il controllo dal computer alla stampante. Usa il comando CMD.
CMD numero di file logico
Perciò abbiamo bisogno che
CMD 1
sia costante

Il computer stamperà READY sulla carta. L'istruzione LIST ora farà sì che il programma venga stampato sulla stampante piuttosto che sullo schermo del monitor.

Ora abbiamo bisogno di PRINT #

PRINT # numero di file logico

per togliere la linea alla stampante, così scriveremo
PRINT # 1

Nota che devi scrivere PRINT # lettera per lettera. Non puoi usare il simbolo "?" che sta per PRINT. Infine devi chiudere il file dopo la stampa:

CLOSE numero logico
Perciò:
CLOSE 1
La nostra sequenza di comandi è:
OPEN 1,4
CMD 1
LIST
PRINT #
CLOSE 1

che corrisponde al comando LLIST disponibile su alcuni microcomputer.
E' possibile stampare informazioni sulla stampante anche durante l'esecuzione di un programma. Devi aprire un file come prima e puoi usare PRINT # nello stesso modo in cui normalmente usi PRINT per lo schermo.
Per esempio:

40 PRINT#1, "GIOCO DEL SOTTOMARINO"

farà sì che "GIOCO DEL SOTTOMARINO" appaia sulla stampante.
Non dimenticare CLOSE 1, alla fine della parte da scrivere con la stampante, che chiude il file.

Su alcune versioni del BASIC è disponibile il comando LPRINT, che ha la stessa funzione.

Queste istruzioni riguardano la manipolazione di file su nastro o su disco, illustrata nei paragrafi successivi.

10.3 File sequenziali

Abbiamo usato il termine "file" parecchie volte in questo corso, per indicare "una collezione di elementi di dati".
Comunque, quando un programma viene memorizzato, diciamo che ha un nome di file e pensiamo sia ai programmi che ai dati come a file.

Dati

Precedentemente abbiamo immesso dati dalla tastiera nel corso dell'esecuzione di un programma, oppure abbiamo letto delle istruzioni DATA che facevano parte del programma. In differenti occasioni abbiamo fatto sì che un programma manipolasse differenti insiemi di dati sovrascrivendo o sostituendo nuove istruzioni DATA alle vecchie.
Quest'ultimo metodo può essere abbastanza efficace per piccoli sistemi che non hanno la possibilità di manipolare file. Ma se vogliamo gestire collezioni di dati di dimensioni considerevoli, abbiamo bisogno di immagazzinare dati su nastro o su disco. Dobbiamo poter immagazzinare nuovi dati su un file in una cassetta o un disco e leggere i dati di tali file per riutilizzarli nell'esecuzione dei programmi.

Incompatibilità di sistemi

La maggior parte dei sistemi di microcomputer differiscono l'uno dall'altro per piccoli dettagli sulla gestione dei file. Cercheremo di concentrarci sui principi generali. I nostri esempi saranno i più semplici possibili, ma saranno riferiti al Commodore 64.

Files sequenziali

Per semplificare considereremo solo il problema di file sequenziali. "File sequenziali" sono quelli in cui leggiamo dal file ogni elemento dei dati nella sequenza in cui è stato creato originariamente. Questo è, naturalmente, l'unico tipo di file che puoi usare su un sistema di registrazione a nastri, poiché un nastro deve essere letto in modo sequenziale.

10.4 Creazione ed accesso

"Creazione" si riferisce all'attività di un programma nello scrivere dei dati dal programma contenuto nella memoria del computer ad un meccanismo periferico: nel nostro caso anche nastro o disco. L'accesso si verifica quando i dati sono letti nella memoria del programma dal meccanismo periferico.
Poiché non possiamo vedere che cosa è conservato sul nastro o sul disco, le attività di creazione ed accesso devono essere complementari. Non puoi sapere se il tuo programma di creazione funziona fino a quando non scrivi un programma di accesso per rileggere i dati. Soltanto allora puoi vederlo sullo

schermo o sulla stampante.
Per rendere questa unità più semplice possibile, terremo separati, quando potremo, la creazione e l'accesso.

Creazione

Per creare un nuovo file o per scrivere su di uno già esistente dobbiamo dire al computer:

- il nome del file,
- che vogliamo aprire il file per scrivere,
- che scriviamo,
- che poi chiudiamo.

Nota che:

- durante la creazione del file, il computer ha bisogno di un numero temporaneo che identifichi tale file;
- fra OPEN e CLOSE il computer è sotto il controllo del registratore.

Un programma di creazione su un Commodore 64 si scrive così:

```
1000 REM **CREA UN FILE**
1010 OPEN 1,1,1,"DATI1"
```




```
1080 CLOSE 1
```

Note

OPEN 1,1,1, ←	dice al computer di aprire un file in uscita
(secondo 1) ←	dice al computer di usare la cassetta di registrazione
(primo 1) ←	il nostro numero interno che useremo per chiamare il file fino a quando resta aperto. Il Commodore 64 ci permette di usare i numeri di file (0–255), ma noi useremo 1 nei nostri esempi.
"DATI1"	il nome del file sulla cassetta o disco su cui verranno immagazzinati i dati
CLOSE1	chiude il file a cui abbiamo dato il numero 1.

Esempio 1a

Scrivi un programma per creare un file di dati di 10 nomi.

Soluzione

```
1000 REM **CREA UN FILE DATI**
1010 OPEN 1,1,1,"DATI1"
1020 REM *****************
1030 PRINT"SCRIVI I DATI (ZZZZ PER SMETTERE)"
1040 INPUT"DATI1";A$
1050 PRINT#1,A$
1060 IF A$<>"ZZZZ" THEN 1040
1070 PRINT"DATI1 E' STATO MEMORIZZATO"
1080 CLOSE 1
```

Programma 1 – Crea un file di dati

Prima di premere RETURN bisogna attivare il registratore

```
DATI1? ALFREDO
DATI1? ANDREA
DATI1? CONSUELO
DATI1? DANIELE
DATI1? FRANCESCO
DATI1? LAURA
DATI1? MARCO
```

```
DATI1? MAURO
DATI1? MORENO
DATI1? NICOLETTA
DATI1? ZZZZ
DATI1 E' STATO MEMORIZZATO
READY
```

e qui spegni.

L'unica piccola novità è PRINT #1,A$. Scrive il valore A$ nel file 1 (puoi vedere il numero di file da usare durante il programma).
Alla fine i 10 nomi (ALFREDO, ANDREA, ecc.) sono tutti sul nastro sotto il nome di file "DATI1" e, se spegni il computer, rimarranno lì.

Accesso

Se vuoi utilizzare di nuovo i tuoi dati, hai bisogno di un programma di accesso, che ha la seguente struttura:

```
10 REM **ACCESSO AD UN FILE DI DATI**
30 OPEN 1,1,0,"DATI1"
50 IF A$="ZZZZ" THEN 80
```




```
80 CLOSE 1
```

Note

OPEN 1,1,0 ← dice al computer di aprire il file in input ("0" per input)

- il secondo 1: dice al computer di usare il registratore
- il primo 1: il nostro numero di file interno (al programma)

"DATI1" — il nome del file che il computer sta cercando su nastro o su disco.

IF A$="ZZZZ" — stiamo usando una registrazione fittizia per segnalare la fine del file.

CLOSE 1 — dice al computer di chiudere il file numero 1.

Il programma lavora così:

Esempio 1b

Scrivi un programma per accedere al file "DATI1" che era stato creato nell'es. 1a e stampa i 10 nomi nel file.

```
10 REM **ACCESSO AD UN FILE DI DATI**
20 PRINT CHR$(147)
30 OPEN 1,1,0,"DATI1"
40 INPUT#1,A$
50 IF A$="ZZZZ" THEN 80
60 PRINT A$
70 GOTO 40
80 CLOSE 1
```

Programma 2 – Accesso ad file di dati.

```
RUN ——————————————— il registratore deve
PRESS PLAY ON TAPE  essere attivato (soltanto
OK                  PLAY)
ALFREDO
ANDREA
CONSUELO
DANIELE
FRANCESCO
LAURA
MARCO
MAURO
MORENO
NICOLETTA
——————————————————— e spegni qui.
READY
```

Ancora una volta abbiamo aperto il file e abbiamo letto i dati:

```
40 INPUT#1,A$
```

Note preliminari agli esercizi

Un serio difetto dei programmi 1 e 2 è che entrambi lavorano soltanto per un file specifico, cioè quello chiamato "DATI1". Ora, se un programma di manipolazione di file deve essere di uso generale, non possiamo riscrivere ogni volta il programma solo per creare o accedere a dati di file con nomi diversi. Introduciamo allora il nome del file in una locazione variabile di memoria, F$, per esempio:

```
INPUT"NOME DEL FILE DI DATI";F$
OPEN 1,1,0,F$
```

Scriviamo dei programmi che risolvano il problema.

Esercizio 1

Modifica il programma 1 in modo da: scrivere una sequenza di nomi su un file, direttamente da tastiera, e dare al file creato un nome variabile.

Esercizio 2

Modifica il programma 2 per accedere al file creato nell'esercizio 1 e stampa il listing.

Esercizio 3

Scrivi un programma che acceda al file creato nell'esercizio 1, cerchi e stampi tutti i nomi che iniziano con la lettera N.

Esercizio 4

Scrivi un programma che acceda al file creato nell'esercizio 1, che introduca i nomi in una lista, e che stampi la lista con gli indici.

10.5 File nei diagrammi di flusso

La soluzione all'esercizio 4 ti dà indicazioni per lo sviluppo di programmi futuri. E' una nostra vecchia amica, di nuovo la lista, che fa tutte le differenze! Avendo ottenuto i dati in forma di lista, possiamo fare qualcosa di più. Prima

di farlo, comunque, proviamo a riassumere la posizione che abbiamo raggiunto.
La soluzione al programma dell'esercizio 1 può essere indicata nella figura 1.

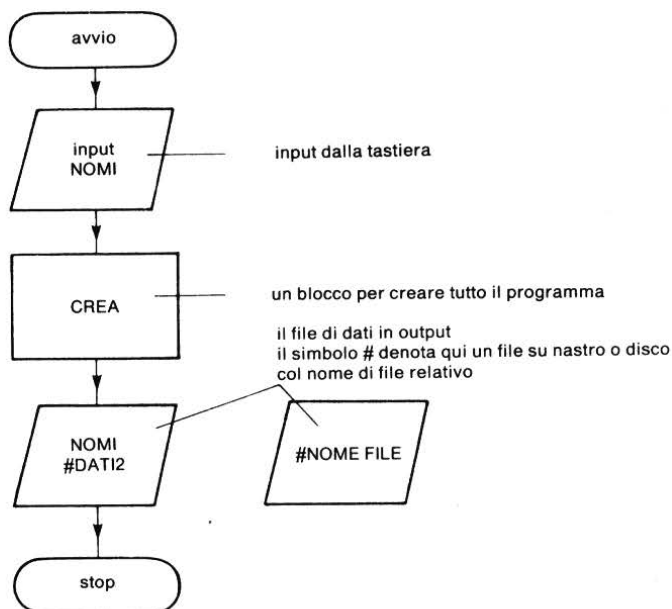


Figura 1 – Diagramma di flusso per creare un file.

La soluzione del diagramma di flusso per l'esercizio 4 viene riportata nella figura 2.

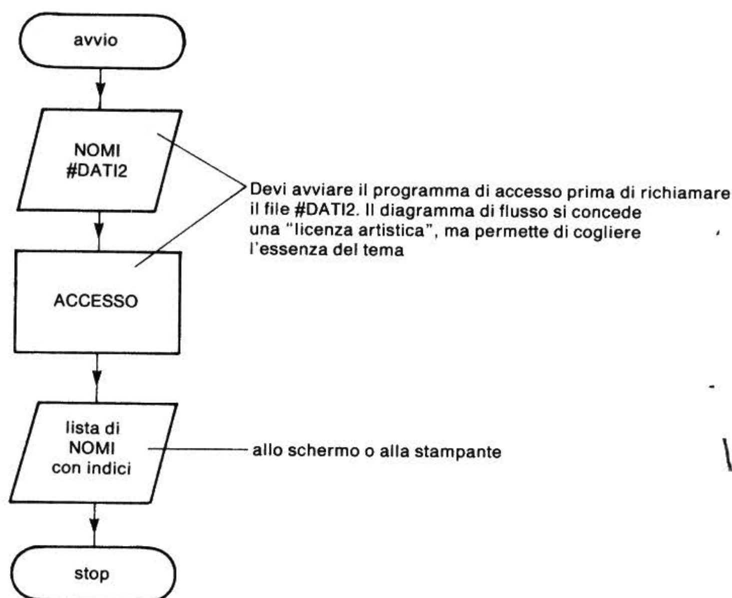


Figura 2 – Accesso ad un file e indici della lista.

10.6 Ordinare dati in un file

La lista di nomi nella risposta all'esercizio 2, deve essere ora riordinata! Abbiamo una routine di accesso e abbiamo sviluppato una routine di ordinamento nell'unità 8. Tutto quello che dobbiamo fare è di legare le due insieme.

Facciamo prima l'algoritmo.



Figura 3 – Accesso e ordinamento.

Il programma consiste semplicemente di tre routine (accesso, ordinamento e stampa), che abbiamo già incontrato, unite insieme.

```
10 REM **ACCESSO AD UN FILE DI DATI**
20 PRINT CHR$(147)
30 DIM X$(50)
```

```
40 C=1
50 INPUT"NOME DEL FILE";G$
60 OPEN 1,1,0,G$
70 INPUT#1,A$
80 X$(C)=A$
90 IF A$="ZZZZ" THEN 120
100 C=C+1
110 GOTO 70
120 CLOSE 1
130 REM **FINE ACCESSO**
140 REM **N E' LA LUNGHEZZA DELLA LISTA**
150 N=C
200 REM ******************
210 REM **ROUTINE DI ORDINAMENTO**
220 FOR K=1 TO N-1
230 FOR L=K+1 TO N
240 IF X$(L)<=X$(K) THEN 280
250 T$=X$(L)
260 X$(L)=X$(K)
270 X$(K)=T$
280 NEXT L
290 NEXT K
300 REM **FINE DELLA ROUTINE DI ORDINAMENTO**
400 REM ******************
410 PRINT"LISTA FINALE ORDINATA"
420 FOR P=1 TO N
430 PRINT P,X$(P)
440 NEXT P
450 PRINT
500 REM ******************
```

routine di accesso (righe 50–130)

routine di ordinamento (righe 200–290)

routine di stampa (righe 410–500)

Programma 3 – Accesso e ordinamento.

```
RUN
NOME DEL FILE? DATI2

LISTA FINALE ORDINATA
 1          ASCOLI
 2          ATALANTA
 3          AVELLINO
 4          BARI
 5          BOLOGNA
 6          CAGLIARI
```

```
7           CAMPOBASSO
8           CESENA
9           FIORENTINA
10          GENOA
11          INTENAZIONALE
12          JUVENTUS
13          LAZIO
14          MILAN
15          NAPOLI
16          PERUGIA
17          PESCARA
18          PISA
19          ROMA
20          SAMPDORIA
21          SPAL
22          TORINO
23          UDINESE
24          VERONA.
25          ZZZZ


READY
```

Esercizio 5

Non vogliamo necessariamente stampare la lista, come abbiamo fatto per scopi dimostrativi nel programma 3, ma vorremmo definitivamente memorizzare i dati ordinati in un file.
Modifica il programma 3 per scrivere la lista di dati ordinati in un file, chiamandolo SDATI.
Come controllerai i risultati del programma?

Esercizio 6

Avendo creato un file di dati ordinati, presto o tardi vorrai fare delle ricerche all'interno di quel file rapidamente.
Combina la routine di accesso alla lista del programma 3 con la routine di ricerca per bisezione dell'unità 9 e crea un programma di ricerca all'interno di un file.

Esercizio 7

Scrivi un programma per accedere a un file ed immettere i dati in una tabella. Scegli tu le dimensioni della tabella e i dati.

10.7 Fusione

Abbiamo creato file, li abbiamo letti (accesso), li abbiamo ordinati e abbiamo operato una ricerca all'interno di essi.
Cos'altro c'è da fare? Bene, molto spesso un file di dati con il quale lavoriamo quotidianamente rimane statico per lungo tempo. Generalmente vogliamo aggiungere o cancellare elementi o fare dei cambiamenti. Il resto di questa unità sarà dedicato a considerare come aggiungere o cancellare elementi in e da un file.
Senza dubbio, hai già presente come svolgere questi compiti e speriamo che possa seguire le tue idee. Noi svilupperemo un approccio standard. Per aggiungere elementi ad un file, opereremo una fusione tra due file. Lo hai già fatto, se hai mai giocato a carte. Hai una mano di carte che hai ordinato e che tieni secondo quell'ordine davanti a te.
Scegli un'altra carta e la metti nella posizione appropriata.
Il processo di fusione riguarda un file principale (organizzato secondo un ordine) e una lista di nuovi elementi (anch'essi organizzati secondo lo stesso ordine):

File principale	**Nuovi elementi (=file di lavoro)**
ADDA	
ADIGE	
	ARNO
BORMIDA	
	BREMBO
BRENTA	
	DORA
ISARCO	
	OGLIO
PIAVE	
PO	
SCRIVIA	
	SERIO
TEVERE	
TICINO	

Ma prima di sviluppare un programma per la fusione, dobbiamo anticipare e risolvere un problema: gli estremi del file principale. Sia nella ricerca per bisezione (Unità 9) che nell'esercizio 6 abbiamo ritenuto necessario assicurar-

ci che il file principale avesse degli estremi. Perciò ora sviluppiamo una routine di programma adeguata.

Gli estremi del file principale

La seguente routine prende un file ordinato (G$) e mette "AAAA" come primo elemento. Non abbiamo bisogno di aggiungere "ZZZZ" come ultimo perché lo abbiamo già usato come record fittizio per segnalare la fine del file (programma 1).

```
10 REM **ACCESSO AD UN FILE DI DATI"
20 PRINT CHR$(147)
30 DIM M$(50)
40 C=1
50 M$(C)="AAAA"   ——— elemento inferiore AAAA,
                      prima dell'accesso al file
60 INPUT"NOME DEL FILE DI DATI";G$
70 OPEN 1,1,0,G$
80 C=C+1
90 INPUT#1,A$
100 M$(C)=A$
110 IF A$="ZZZZ" THEN 130 ——— elemento superiore ZZZZ
                              dopo la chiusura del file
120 GOTO 80
130 CLOSE 1
140 REM ******************
150 N=C
160 REM ******************
170 FOR I=1 TO N
180 PRINT I,M$(I)
190 NEXT I
```

Programma 4 – Aggiunta di AAAA ad un file

```
RUN
NOME DEL FILE DI DATI? SDATI 3

1        AAAA  ——— elemento inferiore
2        ADDA  ┐
3        ADIGE │
4        BORMIDA│
5        BRENTA │
6        ISARCO │
7        PIAVE  ├── file originario
8        PO     ┘   SDATI3
```

```
9        SCRIVIA   file originario
10       TEVERE    SDATI3
11       TICINO
12       ZZZZ      elemento superiore

READY
```

Il programma di fusione

La figura 4 illustra il processo completo. Ancora una volta, ci siamo concessi qualche licenza artistica. Il programma in realtà, non ha due punti di partenza ma, come puoi vedere quando viene eseguito, ci sono due punti di entrata abbastanza distinti per i due file in input.
Il punto essenziale da ricordare è che SDATI3 e SDATI4 sono stati riorganizzati in ordine alfabetico prima di essere immessi nel programma. Lo abbiamo fatto utilizzando programmi sviluppati in precedenza in questa unità.

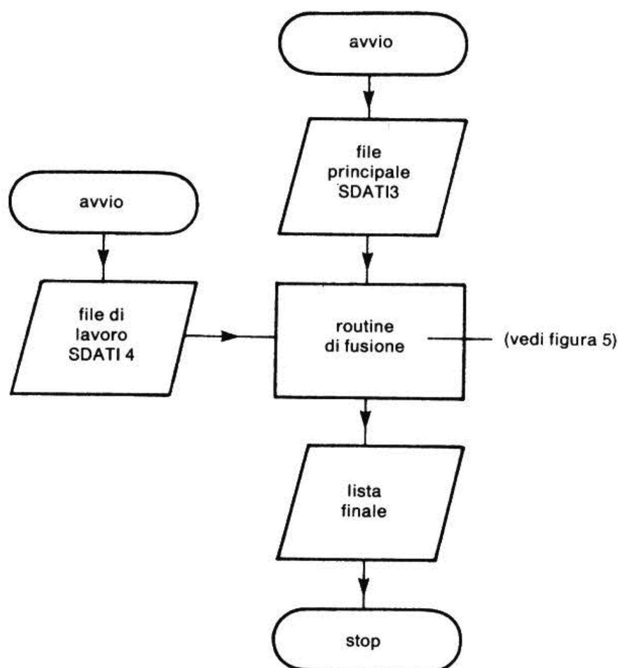


Figura 4 – Schema di routine di fusione.

La routine di fusione

- se l'elemento nel file principale è di ordine minore rispetto all'elemento del file di lavoro
- allora scrivi l'elemento dal file principale nel nuovo file
- altrimenti scrivi l'elemento del file di lavoro nel nuovo file.

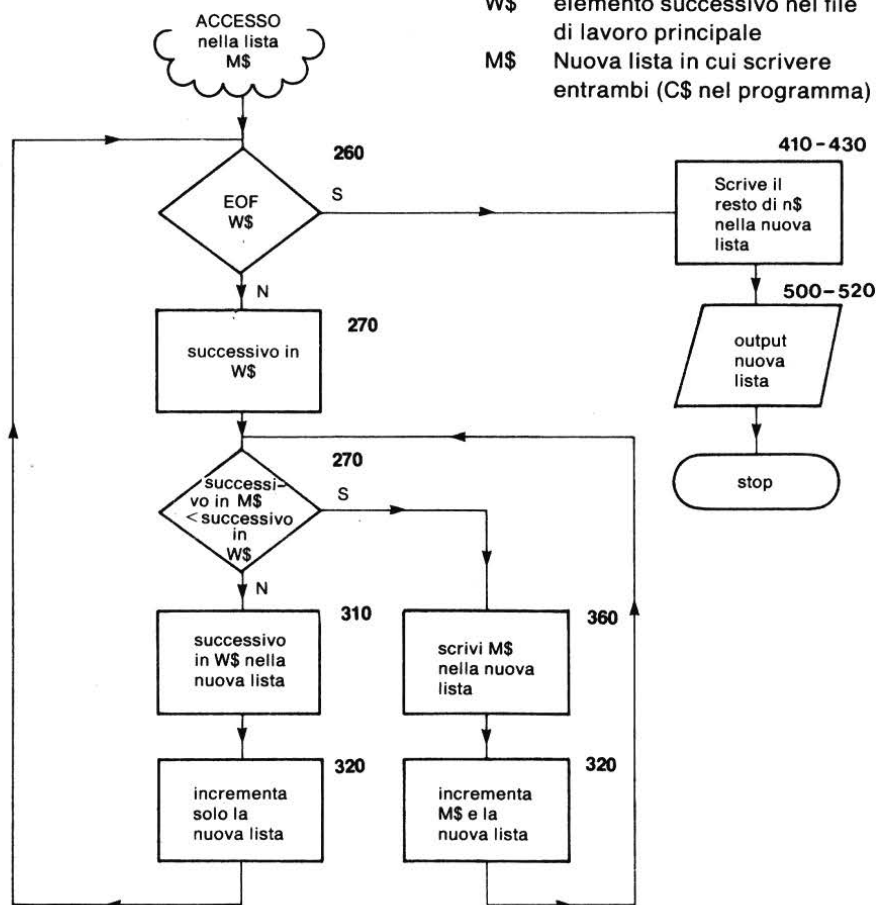


Figura 5 – La routine di fusione in dettaglio.

```
10 REM **ACCESSO AD UN FILE DI DATI**
20 PRINT CHR$(147)
30 DIM M$(50)
35 DIM C$(50)
40 C=1
50 M$(C)="AAAA"
60 INPUT"NOME DEL FILE DI DATI";G$
70 OPEN 1,1,0,G$
80 C=C+1
90 INPUT#1,A$
100 M$(C)=A$
110 IF A$="ZZZZ" THEN 130
120 GOTO 80
130 CLOSE 1
140 REM ******************
150 N=C
160 REM ******************
170 FOR I=1 TO N
180 PRINT I,M$(I)
190 NEXT I
200 REM **FUSIONE**
210 I=1
220 K=1
230 INPUT"NOME DEL FILE DI LAVORO";H$
240 OPEN 1,1,0,H$
250 INPUT#1,W$
260 IF W$="ZZZZ" THEN 350
270 IF M$(I)<W$ THEN 310
280 C$(K)=W$
290 K=K+1
300 GOTO 250
310 C$(K)=M$(I)
320 K=K+1
330 I=I+1
340 GOTO 270
350 CLOSE 1
360 C$(K)=M$(I)
370 K=K+1
380 I=I+1
390 IF I<=N THEN 360
400 REM ******************
500 FOR L=2 TO K-2
510 PRINT L,C$(L)
```

C$ sarà la nuova lista (riga 35)

accesso al file principale, **M$(C)** e inserimento di AAAA e **ZZZZ** (righe 40–130)

stampa del file principale prima della fusione (righe 170–190)

fusione (righe 210–390)

K è troppo alto alla 390 e non vogliamo "ZZZZ"; quindi K–2 (riga 500)

```
520 NEXT L
550 FOR L=2 TO K-2
```

Programma 5 – Il programma di fusione completo

Esecuzione del programma di fusione

Usando prima SDATI3 come file principale:

```
RUN
NOME DEL FILE DATI? SDATI 3

 1          AAAA
 2          ADDA
 3          ADIGE
 4          BORMIDA
 5          BRENTA
 6          ISARCO
 7          PIAVE
 8          PO
 9          SCRIVIA
 10         TEVERE
 11         TICINO
 12         ZZZZ
NOME DEL FILE DI LAVORO? SDATI4
 2          AAAA
 3          ADIGE
 4          ARNO
 5          BORMIDA
 6          BREMBO
 7          BRENTA
 8          DORA
 9          ISARCO
 10         OGLIO
 11         PIAVE
 12         PO
 13         SCRIVIA
 14         SERIO
 15         TEVERE
 16         TICINO

READY
```

Usando, dopo, SDATI4 come file principale:

```
RUN
NOME DEL FILE DATI? SDATI4

 1          AAAA
 2          ARNO
 3          BREMBO
 4          DORA
 5          OGLIO
 6          SERIO
 7          ZZZZ
NOME DEL FILE DI LAVORO? SDATI3

 2          ADDA
 3          ADIGE
 4          ARNO
 5          BORMIDA
 6          BREMBO
 7          BRENTA
 8          DORA
 9          ISARCO
 10         OGLIO
 11         PIAVE
 12         PO
 13         SCRIVIA
 14         SERIO
 15         TEVERE
 16         TICINO

READY
```

Abbiamo eseguito il programma usando i file SDATI3 e SDATI4 come file principale. Puoi vedere che in realtà non c'è il problema di quale file venga chiamato principale e quale di lavoro (è interessante notare che l'immissione del primo dei due (SDATI4) occupa un po' meno spazio nella memoria).
Se volessimo la nuova lista in un file di dati, forse dovremmo aggiungere una routine di creazione alla fine del programma, al posto delle linee 500–520.

10.8 Cancellazione

La cancellazione può sembrare un processo molto diverso dall'addizione e

dalla fusione ma può essere portato a termine modificando il programma di fusione soltanto un po'.
Per prima cosa mettiamo insieme gli elementi da cancellare in un file di lavoro.

Allora:

- se l'elemento nella lista principale = quello nella lista di lavoro, non scrivere l'elemento della lista principale nella nuova lista;
- altrimenti scrivi l'elemento della lista principale nella nuova lista.

Non illustreremo qui il diagramma di flusso relativo a questo caso, ma speriamo che tu possa seguire i cambiamenti del programma di fusione come mostrato dal programma di cancellazione. In questo caso, se rifletti un momento, ti convincerai che le liste di lavoro e principale non sono più intercambiabili.

File principale(SDATI)	**File di lavoro (SDATI5) = da cancellare**
BULLONI	
CACCIAVITI	
CALIBRO	
CHIAVI	
CHIODI	
DADI	
FORBICI	
GUARNIZIONI	
LIMA	CHIAVI
MARTELLO	FORBICI
METRO	NASTRO ISOLANTE
NASTRO ISOLANTE	STAGNO
PAPPAGALLO	TRAPANO
PINZE	
PUNTERUOLO	
RONDELLE	
SALDATORE	
SQUADRA	
STAGNO	
TASSELLI	
TESTER	
TRAPANO	
TRONCHESINO	
VITI	

```
10 REM **ACCESSO AD UN FILE DI DATI**
20 PRINT CHR$(147)
30 DIM M$(50)
35 DIM C$(50)
40 C=1
50 M$(C)="AAAA"
60 INPUT"NOME DEL FILE DI DATI";G$
70 OPEN 1,1,0,G$
80 C=C+1
90 INPUT#1,A$
100 M$(C)=A$
110 IF A$="ZZZZ" THEN 130
120 GOTO 80
130 CLOSE 1
140 REM ******************
150 N=C
160 REM ******************
170 FOR I=1 TO N
180 PRINT I,M$(I)
190 NEXT I
200 REM **CANCELLA**
210 I=1
220 K=1
230 INPUT"NOME DEL FILE DI LAVORO";H$
240 OPEN 1,1,0,H$
250 INPUT#1,W$
260 IF W$="ZZZZ" THEN 370
270 IF M$(I)=W$ THEN 350
310 C$(K)=M$(I)
320 K=K+1
330 I=I+1
340 GOTO 270
350 I=I+1
360 GOTO 250
370 CLOSE 1
380 C$(K)=M$(I)
390 K=K+1
400 I=I+1
410 IF I<=N THEN 380
420 REM ******************
430 FOR L=2 TO K-2
440 PRINT L,C$(L)
450 NEXT L
```

se l'elemento principale = l'elemento di lavoro non fare niente, cerca solo il prossimo elemento principale, altrimenti scrivi l'elemento principale nella nuova lista.

```
500 FOR L=2 TO K-2
510 PRINT L,C$(L)
520 NEXT L
550 FOR L=2 TO K-2
```

Programma 6 - Accesso a datafile.

```
RUN
NOME DEL FILE DATI? SDATI

 1          AAAA
 2          BULLONI
 3          CACCIAVITI
 4          CALIBRO
 5          CHIAVI
 6          CHIODI
 7          DADI
 8          FORBICI
 9          GURNIZIONI
 10         LIMA
 11         MARTELLO
 12         METRO
 13         NASTRO ISOLANTE
 14         PAPPAGALLO
 15         PINZE
 16         PUNTERUOLO
 17         RONDELLE
 18         SALDATORE
 19         SQUADRA
 20         STAGNO
 21         TASSELLI
 22         TESTER
 23         TRAPANO
 24         TRONCHESINO
 25         VITI
 26         ZZZZ
NOME DEL FILE DI LAVORO? SDATI5

 2          BULLONI
 3          CACCIAVITI
 4          CALIBRO
 5          CHIODI
 6          DADI
 7          GUARNIZIONI
 8          LIMA
```

lista principale

```
9           MARTELLO
10          METRO
11          PAPPAGALLO      ——— lista nuova...la vecchia principale
12          PINZE               con cinque elementi cancellati
13          PUNTERUOLO
14          RONDELLE
15          SALDATORE
16          SQADRA
17          TASSELLI
18          TESTER
19          TRONCHESINO
20          VITI

READY
```

Compito 10

1. Scrivi un programma per creare un file principale che contenga (in ordine alfabetico) i nomi degli studenti che prendono in prestito i libri di una biblioteca, registrandoli nella forma:

```
PERSONA CHE      DATA         TITOLO
PRENDE           DEL          DEL
A PRESTITO       PRESTITO     LIBRO
```

   (a) scrivi una routine di fusione delle nuove registrazioni con questo file principale
   (b) scrivi una routine di accesso e ricerca nel file quei libri di cui è scaduto il prestito
2. Scrivi un programma di accesso ad una tabella di valori numerici, completa la somma delle righe e delle colonne e crea un nuovo file per includere queste informazioni extra.

Obiettivi dell'unità 10

Controlla se sei in grado di scrivere programmi in una forma adattabile al tuo microcomputer per:

Creare un file di dati ☐
Accedere ad un file di dati ☐
Ordinare ☐
Aggiungere nuovi elementi ad un file di dati (fusione) ☐
Cancellare elementi da un file di dati. ☐

Risposte agli esercizi

Esercizio 1

```
1000 REM **CREARE UN FILE DI DATI**
1010 INPUT"NOME DEL FILE DI DATI";F$
1020 OPEN 1,1,1,F$
1030 REM **IMMETTE DATI**
1040 PRINT"BATTI I DATI (ZZZZ PER FINIRE)"
1050 INPUT"NOME SUCCESSIVO";N$
1060 PRINT#1,N$
1070 IF N$<>"ZZZZ" THEN 1050
1080 PRINT F$;" E' STATO CONSERVATO"
1090 CLOSE 1
```

Programma 7

```
RUN
NOME DEL FILE DI DATI? DATI2
BATTI I DATI (ZZZZ PER FINIRE)
NOME SUCCESSIVO? CALIBRO
NOME SUCCESSIVO? SQUADRA
NOME SUCCESSIVO? MARTELLO
NOME SUCCESSIVO? TRONCHESINO
NOME SUCCESSIVO? CHIODI
NOME SUCCESSIVO? VITI
NOME SUCCESSIVO? PAPPAGALLO
NOME SUCCESSIVO? BULLONI
NOME SUCCESSIVO? NASTRO ISOLANTE
NOME SUCCESSIVO? TASSELLI
NOME SUCCESSIVO? DADI
NOME SUCCESSIVO? CHIAVI
NOME SUCCESSIVO? GUARNIZIONI
NOME SUCCESSIVO? PINZE
NOME SUCCESSIVO? PUNTERUOLO
NOME SUCCESSIVO? FORBICI
NOME SUCCESSIVO? SALDATORE
NOME SUCCESSIVO? TESTER
NOME SUCCESSIVO? LIMA
NOME SUCCESSIVO? CACCIAVITI
NOME SUCCESSIVO? RONDELLE
NOME SUCCESSIVO? STAGNO
NOME SUCCESSIVO? TRAPANO
```

```
NOME SUCCESSIVO? METRO
NOME SUCCESSIVO? ZZZZ
DATI2 E' STATO CONSERVATO
READY
```

Esercizio 2

```
10 REM **ACCESSO AD UN FILE DI DATI**
20 PRINT CHR$(147)
30 INPUT"NOME DEL FILE DI DATI";G$
40 OPEN 1,1,0,G$
50 INPUT#1,A$
60 IF A$="ZZZZ" THEN 90
70 PRINT A$
80 GOTO 50
90 CLOSE 1
```

Programma 8

```
RUN
NOME DEL FILE DI DATI? DATI2

CALIBRO
SQUADRA
MARTELLO
TRONCHESINO
CHIODI
VITI
PAPPAGALLO
BULLONI
NASTRO ISOLANTE
TASSELLI
DADI
CHIAVI
GUARNIZIONI
PINZE
PUNTERUOLO
FORBICI
SALDATORE
TESTER
LIMA
CACCIAVITI
RONDELLE
STAGNO
```

```
TRAPANO
METRO

READY
```

Esercizio 3

```
10 REM **ACCESSO AD UN FILE DI DATI**
20 PRINT CHR$(147)
30 INPUT"NOME DEL FILE DI DATI";G$
40 OPEN 1,1,0,G$
50 INPUT#1,A$
60 IF A$="ZZZZ" THEN 100
70 IF LEFT$(A$,1)<>"N" THEN 50
80 PRINT A$
90 GOTO 50
100 CLOSE 1
```

Programma 9

```
RUN
NOME DEL FILE DATI? DATI2

NASTRO ISOLANTE

READY
```

Esercizio 4

```
10 REM **ACCESSO AD UN FILE DI DATI**
20 PRINT CHR$(147)
30 DIM B$(50) ---------------------- dimensiona la lista
40 C=1 ----------------------------- inizializza l'indice
50 INPUT"NOME DEL FILE DI DATI";G$    per la lista
60 OPEN 1,1,0,G$
70 INPUT#1,A$
80 B$(C)=A$
90 IF A$="ZZZZ" THEN 120
100 C=C+1 -------------------------- incrementa solo se
110 GOTO 70                          non trova EOF
120 CLOSE 1
130 N=C
```

```
140 FOR I=1 TO N-1
150 PRINT I,B$(I)
160 NEXT I
```

il contatore C indica così la lunghezza della lista

Programma 10

```
RUN
NOME DEL FILE DATI? DATI2

 1          CALIBRO
 2          SQUADRA
 3          MARTELLO
 4          TRONCHESINO
 5          CHIODI
 6          VITI
 7          PAPPAGALLO
 8          BULLONI
 9          NASTRO ISOLANTE
 10         TASSELLI
 11         DADI
 12         CHIAVI
 13         GUARNIZIONI
 14         PINZE
 15         PUNTERUOLO
 16         FORBICI
 17         SALDATORE
 18         TESTER
 19         LIMA
 20         CACCIAVITI
 21         RONDELLE
 22         STAGNO
 23         TRAPANO
 24         METRO

READY
```

Esercizio 5

Invece della routine di stampa alle linee 410–500 del programma 3, dobbiamo scrivere una routine di creazione (linee da 300 a 500 nel successivo programma).

```
10 REM **ACCESSO AD UN FILE DI DATI**
20 PRINT CHR$(147)
30 DIM B$(50)
40 C=1
50 INPUT"NOME DEL FILE DI DATI";G$
60 OPEN 1,1,0,G$
70 INPUT#1,A$
80 X$(C)=A$
90 IF A$="ZZZZ" THEN 120
100 C=C+1
110 GOTO 70
120 CLOSE 1
130 REM **FINE DELL'ACCESSO**
140 REM **E' LA LUNGEZZA DELLA LISTA**
150 N=C
200 REM ******************
210 REM **ROUTINE DI ORDINAMENTO**
220 FOR K=1 TO N-1
230 FOR L=K+1 TO N
240 IF X$(L)>=X$(K) THEN 280
250 T$=X$(L)
260 X$(L)=X$(K)
270 X$(K)=T$
280 NEXT L
290 NEXT K
300 REM **FINE DELLA ROUTINE DI ORDINAMENTO**
400 REM ******************
410 REM **CREA UN FILE DI DATI**
420 INPUT"NOME PER IL FILE DI DATI";F$
430 OPEN 1,1,1,F$
440 FOR P=1 TO N
450 A$=X$(P)
460 PRINT#1,A$
470 NEXT P
480 CLOSE 1
490 PRINT F$;" E' STATO CONSERVATO"
500 REM ******************
```

Programma 11

```
RUN
NOME DEL FILE DI DATI? SDATI
NOME PER IL FILE DI DATI? SDATI
SDATI È STATO CONSERVATO
```

Facciamo una prova di funzionamento con il file DATI2.

```
RUN
NOME DEL FILE DATI? DATI2

BULLONI
CACCIAVITI
CALIBRO
CHIAVI
CHIODI
DADI
FORBICI
GUARNIZIONI
LIMA
MARTELLO
METRO
NASTRO ISOLANTE
PAPPAGALLO
PINZE
PUNTERUOLO
RONDELLE
SALDATORE
SQUADRA
STAGNO
TASSELLI
TESTER
TRAPANO
TRONCHESINO
VITI

READY
```

Esercizio 6

```
10 REM **ACCESSO PER LA RICERCA**
20 PRINT CHR$(147)
30 DIM N$(50)
40 C=1
50 PRINT"PROGRAMMA DI LETTURA SU CASSETTA"
60 INPUT"NOME DEL FILE DI DATI";G$
70 OPEN 1,1,0,G$
```

accesso

```
80 PRINT"FILE APERTO"
90 REM **LETTURA**
100 INPUT#1,A$
110 LET N$(C)=A$
120 IF A$="ZZZZ" THEN 150
130 C=C+1
140 GOTO 90
150 CLOSE 1
160 REM **FINE ACCESSO**
170 REM **N E' LA LUNGHEZZA DELLA LISTA**
180 N=C
190 REM ******************
200 INPUT"NOME RICHIESTO";A$
210 REM **INIZIO RICERCA**
220 L=1
230 H=N
240 IF H-L=1 THEN 500
250 M=INT((L+H)/2)
260 IF Q$=N$(M) THEN 320
270 IF Q$<N$(M) THEN 300
280 L=M
290 GOTO 240
300 H=M
310 GOTO 240
320 REM **FINE DELLA RICERCA**
330 PRINT"SI ";Q$;" E' NELLA LISTA"
350 GOTO 600
500 PRINT Q$;" NON E' NELLA LISTA"
600 PRINT"FINE DELLA RICERCA"
```

ricerca per bisezione

routine di stampa

Programma 12

```
RUN
PROGRAMMA DI LETTURA SU CASSETTA
NOME DEL FILE DI DATI? DATI2

NOME RICHIESTO? METRO
SI METRO E' NELLA LISTA
FINE DELLA RICERCA

READY
```

file di dati ordinato (non possiamo usare la ricerca per bisezione in un file non ordinato)

```
RUN
PROGRAMMA DI LETTURA SU CASSETTA
NOME DEL FILE DI DATI? DATI2

NOME RICHIESTO? CESOIE
CESOIE NON E' NELLA LISTA
FINE DELLA RICERCA

READY
```

Nota: abbiamo visto il programma di accesso con il programma di ricerca per bisezione dell'Unità 9 e funziona. Per rendere il programma assolutamente sicuro, comunque, dobbiamo fare attenzione ai limiti delle liste: ricorda N$(1) = "AAAA" e N$(N) = "ZZZZ" nell'unità 9! Abbiamo affrontato il problema nel programma 5.

Esercizio 7

```
10 REM **ACCESSO AD UN FILE DI DATI**
20 PRINT CHR$(147)
30 DIM A$(30,30)
40 INPUT"N. DI RIGHE E DI COLONNE";R,C
50 INPUT"NOME DEL FILE DI DATI";G$
60 OPEN 1,1,0,G$
70 FOR I=1 TO R
80 FOR J=1 TO C
90 INPUT#1,B$
100 A$(I,J)=B$
110 IF B$="ZZZZ" THEN 140
120 NEXT J
130 NEXT I
140 CLOSE 1
150 REM ******************
160 FOR I=1 TO R
170 FOR J=1 TO C
180 PRINT TAB(10*(J-1));A$(I,J);
190 NEXT J
200 PRINT
210 NEXT I
220 END
```

Programma 13

Run on SDATI

```
RUN
N. DI RIGHE E DI COLONNE? 8,3
NOME DEL FILE DI DATI? DATI2

BULLONI      CACCIAVITI  CALIBRO
CHIAVI       CHIODI      DADI
FORBICI      GUARNIZIONI LIMA
MARTELLO     METRO       NASTRO ISOLANTE
PAPPAGALLO   PINZE       PUNTERUOLO
RONDELLE     SALDATORE   SQUADRA
STAGNO       TASSELLI    TESTER
TRAPANO      TRONCHESINO VITI

READY
```

PARTE 2

GEOS

IL GEOS

Il sistema operativo GEOS è un programma orientato verso l'utente, in quanto la comunicazione con il Commodore 64 è stata notevolmente semplificata.
Per arrivare a tale risultato, gran parte del sistema operativo residente nel computer (Kernal), è stato riscritto; inoltre si è dovuto aumentare notevolmente la velocità di comunicazione con il drive 1541 (4-5 volte), in quanto altrimenti non si sarebbe potuto ottenere un risultato commercialmente apprezzabile.
In tal modo è stato realizzato un programma estremamente professionale che garantisce un ancor brillante futuro al Commodore 64.
GEOS è un vero e proprio ambiente di lavoro, in quanto le applicazioni inserite possono trasmettersi facilmente testi e immagini, risolvendo semplicemente un vecchio problema che tanti grattacapi ha generato: quello della compatibilità.
Il dischetto che avete ricevuto è detto disco di sistema GEOS: tenetelo con cura in quanto è l'unico con cui potrete caricare tutto il sistema.
Questa appendice è stata scritta in modo estremamente semplice, onde facilitare l' approccio con GEOS. Noi vi consigliamo di leggerla con il vostro Commodore di fianco: potrete così esercitarvi mano a mano che vi illustreremo gli strumenti e i comandi. Vi sarà così più immediato il significato delle varie procedure.
A una generica introduzione agli strumenti del GEOS, seguirà una dettagliata relazione su GEOPAINT, GEOWRITE e sugli accessori che avrete a disposizione.

CAPITOLO 1

COMANDI FONDAMENTALI

Cominciamo subito a utilizzare il sistema operativo GEOS; prendete il dischetto "GEOS", inseritelo nel drive e impostate i comandi seguenti:

load "geos",8
run

Dopo qualche secondo apparirà l'immagine in fig. 1.1. Questa rappresenta una specie di menu principale da cui si può:

— accedere alle varie applicazioni che lavorino in ambiente GEOS; sul vostro dischetto ne trovate due GEOPAINT e GEOWRITE

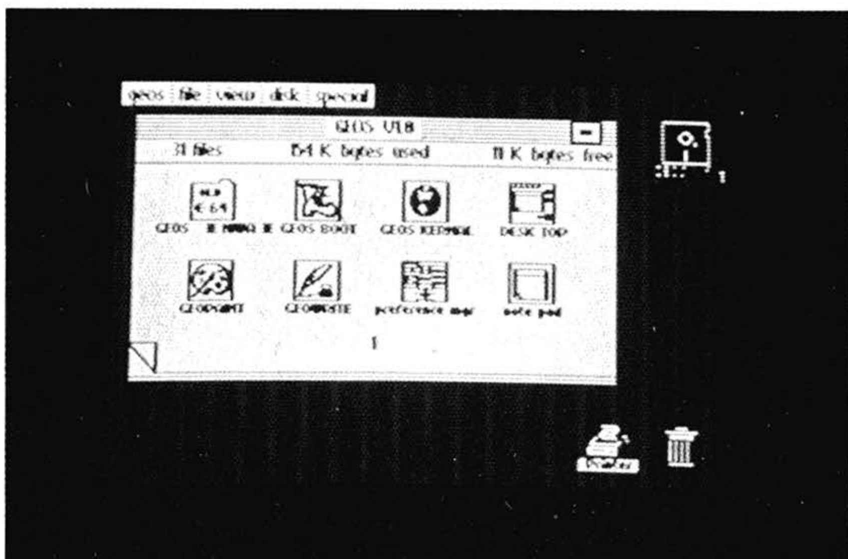


Figura 1.1 - Schermata relativa al Desktop.

— accedere a programmi Basic scritti da voi
— accedere a strumenti ausiliari messi a disposizione dal GEOS, e cioè una calcolatrice ,un blocco di appunti e così via
— infine potrete gestire i file contenuti nel dischetto inserito cancellandoli, copiandoli e così via.

Questa schermata iniziale viene chiamata DESKTOP e verrà richiamata ogni volta che uscirete da una applicazione.
A questo punto inserite il joystick nella porta di controllo 1, e provate a muovere la manopola: la freccia visibile sullo schermo seguirà i vostri movimenti. Essa è l'unico modo per comunicare al GEOS quale operazione effettuare; il suo uso è però molto semplice e intuitivo.
Anzitutto provate a muoverla verso la parte superiore sinistra dello schermo e puntatela sulla parola "geos"; quindi schiacciate il tasto di fire e vedrete apparire il menu rappresentato in fig. 1.2 (pull-down menu). Una volta attivato tale menu si può selezionare l'opzione voluta semplicemente muovendo la freccia sulla parola corrispondente e schiacciando il pulsante di fire; il menu viene così pure disattivato.

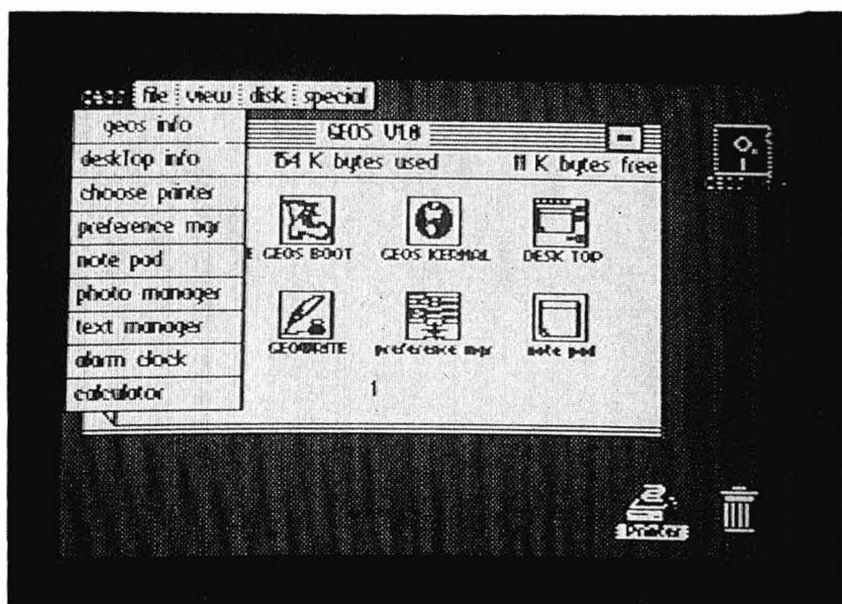


Figura 1.2 - Pull-down menu in funzione.

Ad esempio provate a selezionare l'opzione "geos info": vi compariranno una serie di informazioni sul GEOS. Per ritornare al desktop schiacciate il tasto di fire.
Abbiamo così imparato ad attivare i menu superiori.

Passiamo ora all'immagine centrale sullo schermo: questa rappresenta una finestra, cioè una zona grafica dedicata dal GEOS al programma attualmente attivo. Questo significa che ogni volta che è necessario visualizzare dei messaggi, dei testi o delle immagini, il sistema operativo deve prima riservare una zona dello schermo: lì verrà svolto tutto il lavoro.
In questo caso la finestra rappresenta il dischetto in uso, informandovi sul suo nome, sullo spoazio disponibile e così via.
Nella parte superiore si trova l'intestazione, cioè il nome del disco attualmente inserito; in questo caso GEOS più la versione. Sulla stessa riga ma all'estrema destra si trova un quadrato utile per chiudere il disco attualmente inserito nel drive.
Sulla linea inferiore si trovano informazioni relative al disco presente, e cioè il numero di file, la memoria occupata e quella libera.
Al disotto si trova un cosidetto taccuino del direttorio su cui sono segnati i vari file presenti sul dischetto.
Tale denominazione è giustificata dal fatto che si dispongono di al massimo di otto pagine il cui numero è segnato in basso al centro. Queste possono essere fatte scorrere puntando la freccia sull'angolo in basso a sinistra e schiacciando il bottone di fire; selezionando la parte superiore o inferiore dell'angolo si otterrà la visualizzazione della pagina seguente o precedente.
I file sono rappresentati mediante immagini dette icone. Queste rappresentano indicativamente il tipo di lavoro svolto dall'applicazione; su una pagina ne sono rappresentate otto. Le icone possono essere "attivate": basta puntare sopra la freccia e schiacciare il bottone di fire; noterete il cambiamento di colore dell'immagine: ciò è caratteristico di ogni attivazione. A questo punto opzioni selezionate dai menu superiori riguarderanno il file prescelto.

Schiacciando il bottone di fire due volte di seguito, lasciando in mezzo una pausa di un secondo, apparirà una immagine ulteriore rappresentante i contorni dell'icona; tale elemento potrà essere "trascinato" in giro per lo schermo e rilasciato premendo il tasto di fire. Lo scopo di tale operazione è generalmente quello di copiare un file, di cancellarlo portandolo nel cestino o di stamparlo portandolo sulla stampante. Vedremo in seguito in dettaglio come effettuare tali operazioni.
Provate quindi a cancellare il file "GEOWRITE". Il GEOS vi comunicherà che tale operazione non è possibile in quanto esso è protetto.

Per ritornare al Desktop puntate la freccia sulla casella contenente la scritta O.K. e schiacciate il tasto di fire. Quest'ultimo modo di selezionare opzioni è molto frequente. In particolare vi appariranno dei riquadri in cui vengono presentate alcuni possibili scelte (riquadri di dialogo) : voi puntate su quella che vi interessa e azionate il tasto di fire.

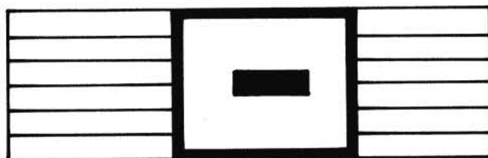


Figura 1.3 - Particolare della finestra contenente il taccuino del direttorio; posizionando la freccia sul quadratino e premendo il tasto di fire, potrete chiudere il disco.

Ad esempio provate a schiacciare due volte di seguito, cioè senza pausa, sull'icona GEOWRITE; in tal modo potrete accedere al wordprocessor, un'applicazione contenuta sul disco.Questa operazione viene chiamata doppio click. Dopo alcuni secondi vi si presenteranno tre scelte:
— creare un nuovo documento
— aprirne uno già esistente
— ritornare al Desktop (quit)
Per il momento scegliete l'opzione QUIT: di Geowrite ci occuperemo in seguito.
Analizziamo ora l'ultima parte del Desktop relativa alla apertura e chiusura di dei dischetti. Provate a puntare la freccia sul quadratino in alto a destra dell'intestazione della finestra (GEOS più la versione in uso) e schiacciate il tasto di fire (fig 1.3). La finestra si sbiancherà e sul dischetto rappresentato in alto a destra apparirà un punto interrogativo.
Questo significa che il GEOS attende l'inserimento di un nuovo dischetto avendo, per così dire, chiuso quello attualmente inserito. In particolare la zona di memoria contenente tutte le informazioni relative, è stata cancellata e i canali di comunicazione con il drive interrotti.
L'operazione inversa è denominata "apertura" del disco, e procede a caricare tutti i dati ad esso relativi.
Una volta inserito nel drive il dischetto nuovo, è sufficente puntare sull'imagine del floppy in alto a destra e schiacciare il tasto di fire.
La sequenza di apertura e chiusura è necessaria ogni qualvolta si vuole cambiare disco, in modo che il Desktop possa essere aggiornato.
Proviamo ora a inserire al posto del disco GEOS, uno qualunque contenente i vostri programmi scritti in basic.

Dopo averlo aperto, appare una finestra indicante che questo è scritto in un formato non GEOS; l'alternativa è fra renderlo compatibile oppure no. Selezionando la scelta YES farete sì che il file del direttorio venga allungato di un blocco, senza per questo perdere alcun dato.
Ciò non ha molta importanza e potrete comunque scegliere l'opzione "NO" che non modificherà il contenuto del disco.
A questo punto appaiono sulla finestra centrale le icone dei vostri programmi; questi possono essere eseguiti semplicemente operando un doppio click sull'icona scelta.
Potrete così vedere funzionare il vostro programma; per ritornare al desktop inserite il disco del GEOS e schiacciate contemparaneamente i tasti RUN-/STOP e RESTORE.

In seguito useremo la dizione "aprire", anche per i programmi e le applicazioni, intendendo così l'operazione di accesso a questi. Con "chiudere" intenderemo il ritorno al Desktop.

Abbiamo così visto in questo capitolo come usare il joystick per selezionare le operazioni volute. Rimane soltanto da mettere in evidenza la semplicità d'uso di un tale sistema, che rende accessibile a tutti l'uso del computer.

Riassunto delle azioni fondamentali del joystick

Attivazione di un menu superiore (pull down-menu)

- Puntate la freccia su uno dei riquadri superiori e azionate il tasto di fire; apparirà così il menu contenente le opzioni possibili
- Desiderando un menu diverso, è sufficiente portare la freccia al di fuori del medesimo; verrà così disattivato

Selezione di un'opzione da un menu superiore

- Dopo avere attivato il menu superiore corretto, puntate la freccia sull'opzione che vi interessa e premete il tasto di fire

Attivazione di un file

- Puntate la freccia sull'icona del file prescelto e schiacciate il tasto di fire; essa cambierà così colore
- Desiderando un file diverso, portate la freccia al di fuori dell'icona attivata e premete il tasto di fire

Trascinamento di un file

— Puntate la freccia sull'icona del file prescelto e schiacciate due volte di seguito, lasciando una certa pausa, il tasto di fire; in tal modo prima si attiva il file e in seguito apparirà un'immagine supplementare che potrà essere portata ovunque sullo schermo
— Una icona così trascinata può essere depositata nel luogo voluto semplicemente premendo il tasto di fire, quando la sua immagine ha assunto la posizione voluta

Finestre supplementari

Molto spesso il GEOS vi offrirà ,in una finestra supplementare, una serie di possibili alternative: per sceglierne una, puntate la freccia sul riquardo contenente quella che vi interessa e premete il tasto di fire.
Ecco il significato di alcune scritte spesso ricorrenti.

QUIT: se siete nei pasticci (vi siete "persi") questa vi permetterà di tornare al Desktop

O.K.: il GEOS vi sta visualizzando un messaggio (in genere di errore) e per proseguire dovete scegliere tale opzione; a volte può pure indicare una risposta positiva a una domanda

CANCEL: annulla l'operazione in corso e ritorna al Desktop; anche questa può diventare un'ancora di salvezza.

Alcune volte il GEOS vi chiederà di inserire il nome di un file: questo è l'unico caso in cui dovrete usare la tastiera e non dimenticate di premere il tasto di RETURN quando avete finito.

Doppio click

Azione destinata ad aprire un file senza passare per i menu superiori.

— Puntate la freccia sull'icona del file che vi interessa e premete il tasto di fire due volte rapidamente, cioè senza lasciare una pausa;
in caso contrario non fareste che iniziare una operazione di trascinamento: per correggere è sufficiente che premiate il tasto di fire un'altra volta in modo da lasciare tutto come in precedenza.

1.2 I comandi del Desktop in dettaglio

Apertura e chiusura del disco

Un disco può essere aperto in due modi:

— selezionando il dischetto in alto a sinistra
— attivando il menu "disk" e selezionando l'opzione open

In tal modo la finestra centrale viene riempita con le icone corrispondenti ai file contenuti nel direttorio.
Se il disco inserito non fosse stato creato con il GEOS, vi verrà chiesto se renderlo GEOS compatibile; tale operazione costa un blocco (256 bytes) sul dischetto e nessun dato sarà cancellato.
Per chiudere un dischetto si può operare in due modi:

— puntando sul quadratino in alto a destra e azionando il tasto di fire
— selezionando con il menu "disk" l'opzione "close".

Tali operazioni permettono di cambiare il dischetto contenuto nel drive.

Comandi per la gestione del dischetto

Attivando il menu "disk" oltre alle oprazioni di close e open si hanno a disposizione molte altre possibilità:

RENAME : permette di cambiare il nome del disco attualmente aperto;

vi verrà chiesto il nuovo nome del disco: inseritelo e premete il tasto RETURN.

FORMAT : serve a formattare un dischetto;
- — prima di attivare il menu, chiudete il disco attualmente in uso e inserite quello da formattare
- — attivate il menu "disk", selezionate l'opzione format
- — il GEOS vi chiederà di inserire il nome del disco: fatelo mediante la tastiera, senza dimenticare di premere RETURN alla fine

VALIDATE: serve a recuperare quei blocchi che vengono inutilmente mantenuti occupati

ADD DRIVE: permette di aggiungere un drive supplementare B

COPY: permette di copiare il disco attualmente aperto su un disco destinazione, precedentemente formattato; in particolare si può creare un backup di tutto il disco GEOS

Quest'ultima operazione è particolarmente importante in quanto è sempre presente la possibilità di perdere dei programmi sul dischetto GEOS; pertanto vi consigliamo caldamente di creare un disco di backup. Questo però non potrà essere usato per caricare il GEOS e vi servirà come serbatoio di programmi.
Prendete un disco vuoto e formattatelo scegliendo l'opzione "format" dal menu "disk"; scegliete un nome qualunque ma che non sia "GEOS". Terminata la formattazione reinserite il dischetto originale, apritelo (come spiegato all'inizio di questo paragrafo) e selezionate dal menu "disk" l'opzione "copy" ; il programma vi chiederà alternativamente di inserire il dischetto di backup e quello originale finchè la copia non sarà terminata.
A questo punto prendete il disco di backup e mettetelo in un luogo sicuro: se un programma del disco GEOS non dovesse più funzionare lo sostituirete con quello equivalente seguendo le indicazioni del prossimo paragrafo.

Copia di un file

Se volete duplicare un file sullo stesso disco, seguite la procedura seguente:

— attivate l'icona del file interessato
— attivate il menu "file" e selezionate l'opzione "duplicate"
— il GEOS vi chiederà il nuovo nome di questo file, in quanto non ne possono coesistere due sullo stesso dischetto con lo stesso nome; inseritelo mediante la tastiera, non dimenticando di schiacciare il tasto RETURN alla fine.

Per duplicare un file da un disco all'altro dovete:

— trascinare l'icona del file interessato al di fuori del direttorio portandolo nella parte inferiore dello schermo, e depositarla lì

— estrarre dal drive il disco sorgente
— inserire quello destinazione e aprirlo
— trascinare l'icona del file che si trova in basso, dentro il taccuino del direttorio e depositarla nel punto voluto
— seguire le indicazioni del GEOS che vi chiederà di inserire alternativamente il disco sorgente e quello destinazione

Ovviamente il disco destinazione dovrà essere GEOS compatibile. Siccome si possono depositare fino a quattro icone sul fondo, questo è il numero massimo di file che potrete copiare in una volta.

Come mettere in ordine il taccuino del direttorio

Abbiamo visto che il direttorio è rappresentato da un taccuino di al massimo otto pagine su cui sono segnate le icone dei file presenti; queste possono essere spostate a piacere in modo da avere una visualizzazione più vicina alle esigenze dell'utente. Per muovere un'icona sul taccuino dovete:

— trascinare quella del file interessato sul fondo, sotto il taccuino e rilasciarla lì
— selezionare la pagina del taccuino su cui dovrà essere inserita l'icona
— trascinarla all'interno del taccuino, rilasciandola nel punto desiderato.

In tal modo potrete separare i vari file del direttorio, mantenendo il taccuino in ordine.

Comandi per gestire i file

Attivando il menu "file" appaiono alcune opzioni necessarie per gestire i file presenti sul dischetto attualmente aperto.
Prima di selezionarle è però necessario attivare l'icona del file su cui si vuole agire.

OPEN: apre il file attivato; è equivalente al doppio click

DUPLICATE: copia il file attivato sullo stesso disco; il Geos vi chiederà di introdurre un nome nuovo.

RENAME: permette di cambiare il nome al file attivato PRINT : stampa il file attivato; è equivalente a trascinare e rilasciare l'icona sulla stampante; per scegliere la corretta stampante leggere il capitolo dedicato agli accessori del GEOS

GET INFO: visualizza informazioni varie sul file attivato;

Per quanto riguarda quest'ultima opzione, bisogna notare la presenza, sotto la scritta "author", di un quadratino che indica se un file è protetto oppure no (se si ha lo stesso colore dei caratteri) ; può essere cambiato semplicemente puntando la freccia sopra e premendo il tasto di fire. Questo vi permetterà di proteggere i vostri documenti da cancellazioni accidentali, oppure di sproteggere i programmi GEOS: quest'ultima è un'attivtà sconsigliata in quanto potreste cancellare applicazioni importanti come GEOWRITE.
In fondo alla finestra esiste un riquadro su cui potrete scrivere i vostri commenti usando la tastiera.
Per tornare al Desktop dovete togliere la finestra: puntate la freccia sul quadratino in alto a destra della stessa; (a lato del nome del file su cui si leggono le informazioni) e premete il tasto fire.

Visualizzazione del direttorio

Attivando il menu "view" è possibile visualizzare il direttorio con un elenco di nomi in vece che di icone. Bisogna perciò selezionare con quale criterio verranno ordinati i file.

NAME: i file vengono presentati in ordine alfabetico

DATE: vengono presentati in ordine temporale; il file aggiornato più recentemente è scritto per primo

SIZE: vengono presentati in ordine di grandezza, il più grande viene prima

KIND: vengono presentati raggruppati per tipo

È importante notare che un file può essere attivato se e solo se è presente sullo schermo il taccuino del direttorio con le icone.
Per visualizzarlo è sufficiente attivare il menu "view" e selezionare l'opzione "icon".

Ritorno al Basic

Attivando il menu "special" e selezionando l'opzione "BASIC" tornerete al basic, dove potrete scrivere i vostri programmi normalmente. Per ritornare al GEOS dovete:

— inserire nel drive il dischetto GEOS
— premere contemporaneamente i tasti RUN/STOP e RESTORE

Altre opzioni del menu "special" sono:

— RESET ricarica il GEOS

CAPITOLO 2

GEOPAINT

2.1 INTRODUZIONE

GEOPAINT è un potente e flessibile editor grafico che semplifica la stesura di documenti in cui l'uso delle figure è predominante rispetto a quello del testo. Quest'editor, come ogni parte del sistema operativo GEOS, è orientato all'utente finale che non abbia familiarità con alcun sistema di calcolo. Per questo è stato riprodotto un ambiente di lavoro simile a quello in cui ogni disegnatore è abituato a svolgere la propria attività. Ci si troverà così a lavorare con carta, matita, gomma e qualsiasi altro strumento che ogni grafico ha a disposizione.

2.2 IL PRIMO DOCUMENTO CON GEOPAINT

L'ambiente che si incontra dopo aver lanciato il sistema operativo GEOS mostra i file presenti sul dischetto mediante delle icone a cui è associato il nome del file che esse rappresentano. Fra le tante mostrate, vi sarà anche quella su cui appare il disegno di una tavolozza di colori. Questa è l'icona di GEOPAINT. Aprendo con un doppio click questa applicazione, si entrerà nell'ambiente di lavoro più sopra illustrato. Dopo qualche secondo vi si mostrerà la schermata visibile in fig. 2.1. Poichè finora non avete ancora lavorato con GEOPAINT, e quindi non disponete di un documento preesistente, dovrete scegliere l'opzione CREATE che vi mette a disposizione un nuovo foglio da disegno su cui lavorare. Questo sarà in futuro rintracciabile mediante il nome che ora vi viene richiesto da tastiera e che dovete terminare con il tasto RETURN. Se durante la battitura avete commesso qualche errore, potrete correggerlo col tasto INST/DEL purchè non abbiate ancora digitato il tasto RETURN. L'opzione CANCEL che avete sul video non serve per correggere il nome del file, ma provvede ad interrompere l'operazione CREATE che avete intrapreso.
Ora vi trovate al tavolo di disegno con un foglio di carta bianca davanti. Sulla sinistra trovate gli strumenti da disegno che avete a disposizione. Di questi potrete usarne uno solo per volta e quello che adopererete sarà evidenziato

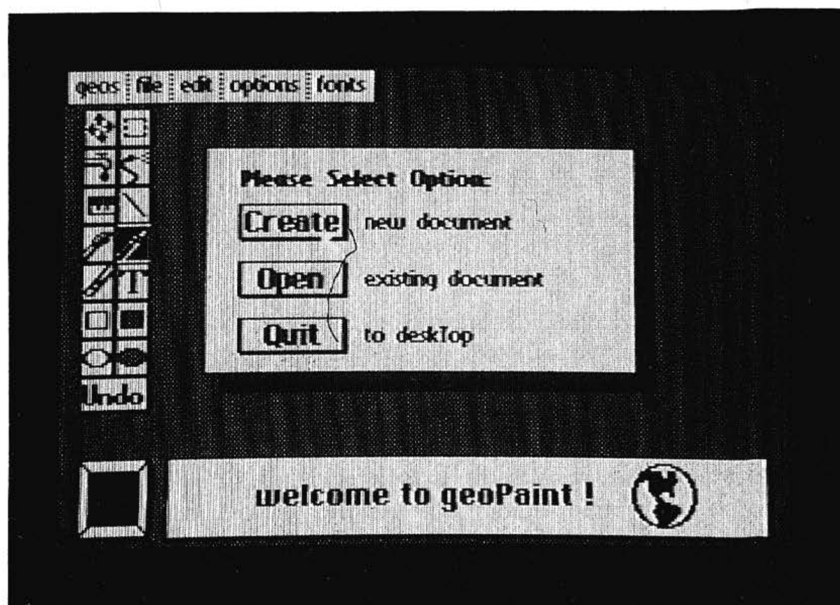


Figura 2.1 - L'editor grafico GEOPAINT.

con un colore di fondo diverso. Poichè siete appena entrati in questo ambiente si assume che stiate usando la MATITA. Questa è d'uso estremamente semplice: se vi trovate già sul foglio da disegno, per iniziare il tracciamento di una linea sarà sufficiente premere il tasto di fire per appoggiare la punta della MATITA sul foglio e quindi muoversi mediante il joystick nella direzione voluta. Azionando nuovamente il tasto di fire, alzeremo la matita dal foglio di disegno e ciò ci permetterà di spostarci liberamente. Come vi sarete accorti il colore della matita cambia a seconda che questa sia appoggiata o sollevata dal foglio.

Per posare lo strumento che abbiamo in mano e prendere quello desiderato, è sufficiente spostare la freccia su quest'ultimo e premere il tasto di fire. Per ora non faremo niente di simile ma ci accontenteremo di imparare ad uscire da questo ambiente per ritornare al DESKTOP. Ciò si otterrà attivando il pull-down menu che contiene la scritta FILE e quindi selezionando l'opzione QUIT. Ora siamo ritornati al DESKTOP e sfogliando il taccuino del directorio troveremo una nuova icona corrispondente al documento su cui abbiamo tracciato una linea.

2.3 GLI STRUMENTI DI GEOPAINT IN DETTAGLIO

MOVIMENTO DELL'AREA GRAFICA

Il foglio su cui lavoriamo è molto più grande di quanto possiamo vedere sul nostro schermo. La parte che ci viene mostrata è detta AREA GRAFICA. Per conoscerne la posizione, dobbiamo guardare nella finestra di stato (STATUS WINDOW) posta nell'angolo in basso a destra del video. Infatti vi troviamo rappresentati sia il foglio da disegno (rettangolo grande) che l'AREA GRAFICA stessa (rettangolo piccolo). Per visualizzare una diversa porzione del nostro documento, dovremo spostare la posizione di quest'ultima mediante lo strumento di lavoro che in fig. 2.2 è indicato con 6. Dopo averlo attivato appariranno in sovraimpressione 4 frecce fra loro ortogonali che indicano i quattro possibili spostamenti selezionabili con il joystick. La finestra di stato ci mostrerà ad ogni istante la posizione dell'AREA GRAFICA all'interno del foglio. Premendo il tasto di fire questo strumento viene deposto e viene riattivato l'ultimo strumento usato.

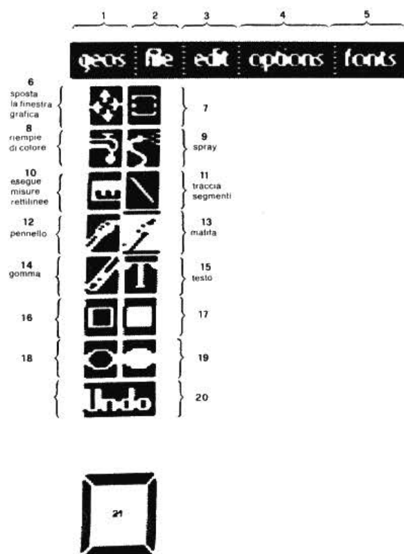


Figura 2.2 - Gli strumenti di lavoro di GEOPAINT.

BOX

Lo strumento, che in fig. 2.2 è indicato con 7, ci permette di definire un rettangolo (BOX) all'interno dell'AREA GRAFICA. Questo ha i lati paralleli al bordo del foglio e pertanto è individuabile con gli estremi di una sua diagonale. La parte di disegno che vi è contenuta può essere mossa, copiata, cancellata e manipolata secondo le varie opzioni che vengono presentate nella finestra di stato all'atto dell'attivazione di BOX.
Vediamone l'uso: dopo averlo attivato, bisogna fissare la posizione e le dimensioni del rettangolo definendo una sua diagonale. Ciò si realizza in due passi:

a) ci si posiziona sul primo estremo di questa e si preme il tasto fire
b) si ripete l'operazione precedente anche per il secondo estremo.

Fatto ciò possiamo selezionare nella finestra di stato l'opzione desiderata. Se questa dovesse essere COPY o MOVE allora dovremo riposizionarci sull'ultimo estremo della diagonale definita e premere il tasto di fire per agganciare alla freccia il rettangolo. Ora, senza fermarsi, si sposterà la freccia nella nuova posizione che si desidera che BOX assuma. Appena la freccia si sarà arrestato, il contenuto di BOX verrà spostato o copiato nella nuova posizione. Premendo nuovamente il tasto di fire si sgancerà la freccia da BOX. Diversamente ogni movimento della freccia verrà trasmesso a BOX.

RUBINETTO

Questo strumento, indicato con 8 in fig. 2.2, provvede a riempire un'area chiusa da linee con il colore correntemente selezionato dalla tavolozza dei colori (vedi). Se viene attivato su un'area aperta l'intera AREA GRAFICA verrà colorata.
Per usare RUBINETTO sarà sufficiente attivarlo, portarsi all'interno dell'area chiusa e premere il tasto di fire.

SPRAY

Questo strumento, indicato in fig. 2.2 con 9, permette di disegnare a spruzzo. L'uso di SPRAY è del tutto simile a quello della MATITA (vedi) ma differisce per il tipo di tratto che realizza.

RIGHELLO

Talvolta sorge la necessità di misurare la distanza fra due punti. RIGHELLO (punto 10 in fig. 2.2) soddisfa a questa necessità.
Per usare questo strumento è necessario che lo selezionate e che vi posizionate su uno dei due punti di cui volete conoscere la distanza. Premete il tasto di fire, posizionatevi sul secondo punto e premetelo nuovamente. La misura desiderata, insieme ad altre informazioni, è mostrata nella finestra di stato. È possibile ottenere questa misura in pixel o in pollici semplicemente attivando nella finestra di stato l'unità di misura desiderata.

TRACCIATORE DI SEGMENTI

Come vi sarete accorti, tracciare con la MATITA delle righe diritte con un'inclinazione qualsiasi, non è un'operzione agevole.
Il TRACCIATORE DI SEGMENTI, che in fig. 2.2 è indicato con 11, permette di risolvere rapidamente questo problema. L'uso di questo strumento è simile al precedente (vedi).

PENNELLO

Come abbiamo visto più sopra, la MATITA ci consente di tracciare con la massima libertà una generica linea. Purtroppo il tratto di questa è fisso.
Il PENNELLO, pur mantenendo le stesse potenzialità e modalità d'uso di MATITA (vedi), ci consente di selezionare in un vasto insieme di tratti predefiniti quello più adatto al disegno. Per questo vi si rimanda al paragrafo in cui vengono trattate le opzioni.

MATITA

Questo strumento, indicato con 13 in fig. 2.2, permette tracciare con la massima libertà delle linee sottili. Se desiderate utilizzarlo selezionatelo e spostatevi nel punto in cui dovete iniziare a tracciare.
Premete il tasto di fire e quindi spostatevi utilizzando il joystick. La punta della matita lascerà un sottile tratto dove passerà. Se desiderate sollevare la matita dal foglio basterà agire nuovamente sul tasto di fire.

GOMMA

Questo strumento (fig. 1 punto 14) permette di cancellare qualsiasi segno indesiderato.
L'uso della GOMMA è simile a quello di MATITA (vedi).

TRASFERIBILI

Talvolta sorge la necessità di corredare un disegno con un testo. I TRASFERIBILI, che in fig. 2.2 sono indicati con 15, soddisfano a questa esigenza. I caratteri disponibili appartengono a diverse serie (FONTS).
Queste possono essere modificate sia selezionando una particolare dimensione del carattere sia utilizzando uno dei vari stili scelto fra quelli disponibili (per esempio bordato, sottolineato, ...). La selezione di una serie di caratteri e delle loro dimensioni sarà trattata nel paragrafo FONTS.
Qui viene solo spiegato l'uso dei TRASFERIBILI e le modalità di scelta di uno stile. Per aggiungere del testo ad un disegno è necessario anzitutto attivare i TRASFERIBILI e successivamente definire il rettangolo, detto FINESTRA DI SCRITTURA, in cui verranno disposti i caratteri digitati da tastiera.
Le modalità di definizione di questo, sono analoghe a quelle descritte nel paragrafo BOX (vedi). Dopo aver definito la FINESTRA DI SCRITTURA apparirà al suo interno un cursore che ci segnala la disponibilità di GEOPAINT ad accettare da tastiera il testo desiderato.
Il tasto RETURN in questo caso non serve per terminare ogni parola composta, ma deve essere usato come un "a capo". Dopo aver digitato qualche parola, si potrà scegliere lo stile più adatto al documento utilizzando le vaie opzioni disponibili nella FINESTRA DI STATO. Fatto ciò si dovrà digitare l'intero testo desiderato, correggendo col tasto INST/DEL gli eventuali errori commessi. Nel caso in cui l'intero testo non possa essere contenuto nella FINESTRA DI SCRITTURA è possibile ridefinirne un'altra secondo le solite modalità. Ciò annulla quanto è stato scritto con lo strumento TRASFERIBILI ma salva il testo composto che riappare nella nuova FINESTRA DI SCRITTURA.

RETTANGOLI VUOTI

Questo strumento, indicato con 16 in fig. 2.2, vi consente di tracciare rapidamente qualsiasi rettangolo. Le modalità d'uso sono quelle consuete che vengono utilizzate nella definizione della BOX (vedi).

RETTANGOLI PIENI

Talvolta è utile disporre di arre rettangolari colorate. Questo risultato può essere facilmente raggiunto utilizzando in combinazione RETTANGOLI VUOTI e RUBINETTO oppure, più rapidamente, lo strumento RETTANGOLI PIENI (punto 17 in fig. 2.2). Il suo uso è simile al precedente (vedi).

COMPASSO

Qualsiasi circonferenza può essere rapidamente tracciata utilizzando lo strumento che in fig. 2.2 è indicato con 18.
L'uso di COMPASSO si articola in due passi: a) ci si posiziona sul centro della circonferrenza da tracciare e si preme il tasto fire. b) ci si spostra lungo un raggio fino a raggiungerne l'estremità e quindi si preme nuovamente il tasto di fire.
Durante questa operazione si osserverà una circonferenza che si modifica dinamicamente in modo da avere come raggio il segmento compreso fra il centro precedentemente definito e la posizione attuale della freccia.

CERCHIO

Questo strumento (fig. 2.2 punto 19) permette di tracciare rapidamente un qualsiasi cerchio. Il suo uso è identico a quello di COMPASSO.

UNDO

Ogni operazione compiuta con gli strumenti sopra descritti non è immediatamente riportata in maniera definitiva sul foglio da disegno. Ogni cambiamento diviene definitivo solo quando se ne intraprende un'altro.
Se prima di apportare una nuova modifica si desidera annullare l'ultima operazione compiuta ci si può utilmente servire dello strumento UNDO. Per utilizzarlo sarà sufficiente attivarlo.

IL COLORE ATTUALE

In fig. 2.2 è indicato con 21 un quadrato riempito con il colore che abbiamo correntemente selezionato. In effetti non si tratta propriamente di un colore

ma di un esempio del tratteggio che si otterrebbe utilizzando lo strumento RUBINETTO.
Per modificare il colore attuale è necessario che ci venga mostrata l'intera tavolozza dei colori disponibili da cui dovremo scegliere. Ciò si ottiene posizionando la freccia all'interno del quadrato più sopra menzionato e premendo il tasto di fire.
Fatto ciò si sceglierà mediante il joystick il colore desiderato.

2.4 I PULL-DOWN MENU

Ciascuno di questi può essere attivato posizionadosi all'interno del rettangolo che contiene il nome che lo individua e premendo il tasto di fire. Vengono così presentate le varie opzioni disponibili. Se fra queste troviamo quella che ci interessa sarà sufficiente posizionarvisi e premere nuovamente il tasto di fire. Diversamente il pull-down menu può essere disattivato semplicemente uscendo dall'area occupata dalle varie opzioni.

FONTS

Come già accennato, GEOPAINT mette a disposizione dell'utente diverse serie di caratteri detti FONTS.
La particolare serie di caratteri desiderata può essere utilizzata selezionandola dal pull-dawn menu FONTS. La scelta di una di queste provoca l'automatica attivazione dello strumento TRASFERIBILI. Quando viene scelto un particolare FONT ci viene anche richiesto di selezionare la dimensione desiderata per ogni carattere della serie. La selezione della dimensione del carattere viene effettuata, come di consueto, premendo il tasto di fire sulla dimensione desiderata.

OPTIONS

PIXEL EDIT Questa opzione consente di effettuare con grade precisione ritocchi su una particolare area del grafico che correntemente abbiamo sul video. Infatti, poichè il disegno su cui operiamo è realizzato accostando una serie di pixel, ossia di puntini, l'opzione PIXEL EDIT ci consente di accedere con facilità a ciascuno di essi; potremo così lavorare su un'immagine ingrandita di una porzione del disegno contenuto nell'AREA GRAFICA.

L'uso di questo strumento è semplice: dopo averlo attivato viene mostrato sul video un rettangolo, detto ZOOM AREA, che si sposterà seguendo i movimenti del joystick. Poichè il disegno ivi contenuto è quello che verrà ingrandito, dovremo posizionare ZOOM AREA sul particolare che ci interessa e successivamente premere il tasto di fire per comunicare a GEOPAINT che quella è l'area selezionata. Subito ci verrà mostrata l'immagine ingrandita. Su questa ora possiamo lavorare con alcuni dei consueti strumenti che GEOPAINT mette a disposizione.
Infatti in modalità PIXEL EDIT non sono disponibile tutti gli attrezzi da disegno.

NORMAL EDIT Questa opzione è quella che ci permette di di uscire dalla modalità PIXEL EDIT. Per attivarla è sufficiente aprire il pull-down menu e selezionarla.

CHANGE BRUSH Lo strumento pennello, come già accennato permette di realizzare diversi tipi di tratti. Quest'opzione serve a selezionare quello più adatto fra i vari disponibili.
Quando CHANGE BRUSH è attivato, secondo le solite modalità, vengono presentati nella finestra di stato i tratti fra cui possiamo scegliere. Quello correntemente selezionato viene evidenziato in un quadrato. Per sostitirlo sarà sufficiente posizionarsi sul tratto desiderato e premere il tasto di fire.

DISPLAY PAGE Poichè sul video ci viene mostrata solo l'AREA GRAFICA che contiene una piccola parte di quanto abbiamo tracciato sul nostro foglio da disegno, facilmente si può avere un sensazione di smarrimento dovuta alla mancanza di una visione di insieme.
DISPLAY PAGE soddisfa a questa esigenza permettendo di rappresentare sul video l'intero documento su cui stiamo lavorando. Per utilizzare questa opzione sarà sufficiente attivarla.
Verrà mostrato l'intero foglio e la scritta OK su cui dovremo posizionarci e premere il tasto di fire per poter continuare a disegnare.

UPDATE FILE Per comprendere il significato di questa opzione e della seguente, è necessario sapere che GEOPAINT tiene nella memoria centrale del calcolatore solo il contenuto dell'AREA GRAFICA. Sul dischetto invece è disponibile l'intero foglio da disegno. UPDATE FILE ci permette di ricopiare su questo le modifiche che abbiamo apportato sul video. Tuttavia è bene precisare che GEOPAINT in certi casi effettua automaticamente questa operazione. Un caso tipico è il movimento dell'AREA GRAFICA mediante l'apposito strumento.

RECOVER FROM FILE L'effetto di questa opzione è opposto a quello della precedente. Infatti, nel caso che siano stati combinati pasticci nell'AREA GRAFICA, RECOVER FROM FILE permette di ripristinare quanto avevamo precedentemente a diposizione su dischetto. Le modalità d'uso si limitano all'attivazione di questa opzione.

EDIT

CUT Questa opzione, come le due seguenti, deve essere usata in combinazione con BOX che definisce un rettangolo all'interno dell'AREA GRAFICA. Infatti CUT toglie dal documento il contenuto di questo e lo pone in un'area di memoria presente su disco detta PHOTO SCRAP. Un nuovo impiegho di CUT provocherà la perdita di del precedente contenuto di PHOTO SCRAP. Pertanto per togliere un particolare dal nostro documento dovremo: a) definire mediante BOX il particolare che intendiamo eliminare. b) attivare l'opzione CUT.

COPY Questa opzione è del tutto simile alla precedente. Vi si differenzia poichè non toglie dal disegno il contenuto di BOX ma lo copia in PHOTO SCRAP.

PASTE A nulla serve depositare o copiare in PHOTO SCRAP particolari di un documento se questi non possono venir riutilizzati altrove. PASTE provvede a copiare il contenuto di PHOTO SCRAP nel BOX che dovremo aver definito prima di attivare questa opzione.

FILE

CLOSE Al termine di una sessione di lavoro (cioè quando abbiamo finito un disegno o comunque quando intendiamo interromperlo) dobbiamo attivare quest'opzione. In risposta ci verrà richiesto, mediante un menu identico a quello che si incontra appena si apre GEOPAINT da DESKTOP, quali azioni intendiamo intraprendere. Possiamo creare un nuovo documento, aprirne uno già preesistente od abbandonare il tavolo da disegno ritornando così a DESKTOP. La scelta fra queste alternative va eseguita mediante il joystick.

PRINT Selezionando questa opzione si manda in stampa l'intero documento corrente. Poichè GEOS supporta diversi tipi di stampanti bisognerà accertarsi che il primo PRINTER DIVER che si incontra sul taccuino direttorio sia quello compatibile con la stampante che si intende utilizzare. Per un ulteriore approfondimento si rimanda all'esame dell'accessorio Choose Printer visibile da DESKTOP.

RENAME Talvolta può essere utile cambiare il nome del documento corrente. RENAME permette di soddisfare a questa necessità. Appena si attiva quest'opzione ci viene richiesto il nuovo nome. Questo, digitato da tastiera e terminato da RETURN, non può superare i 16 caratteri. L'opzione CANCEL che è attivabile mentre si digita il nuovo nome serve ad interrompere l'operazione di RENAME senza che il nome del documento venga cambiato.

QUIT Questa opzione agisce esattamento come la CLOSE ma, senza chiederci alcunchè, ci rimanda a desktop.

GEOS

Per l'esame di quasi tutte le opzioni che sono selezionabili da questo pull-down menu si rimada al capitolo che tratta gli accessosi di GEOS. Qui trattiamo brevemente solo l'opzione GEOPAINT INFO.
Questa, una volta attivata, mostra alcune informazioni riguardanti il programma GEOPAINT.

CAPITOLO 3

L'USO DI GEOWRITE, IL WORDPROCESSOR

3.1 INTRODUZIONE

In questo capitolo tratteremo l'uso della seconda applicazione contenuta sul vostro dischetto e cioè il wordprocessor. Questo è un programma che vi permette di scrivere testi in vari stili e formati, di memorizzarli su dischetto e infine di stamparli. Per cominciare tornate al Desktop e visualizzate la prima pagina del taccuino del direttorio.
Aprite dunque l'applicazione GEOWRITE con i metodi già visti. Dopo alcuni istanti il GEOS vi presenterà tre possibili scelte:
CREATE vi permette di creare un nuovo documento

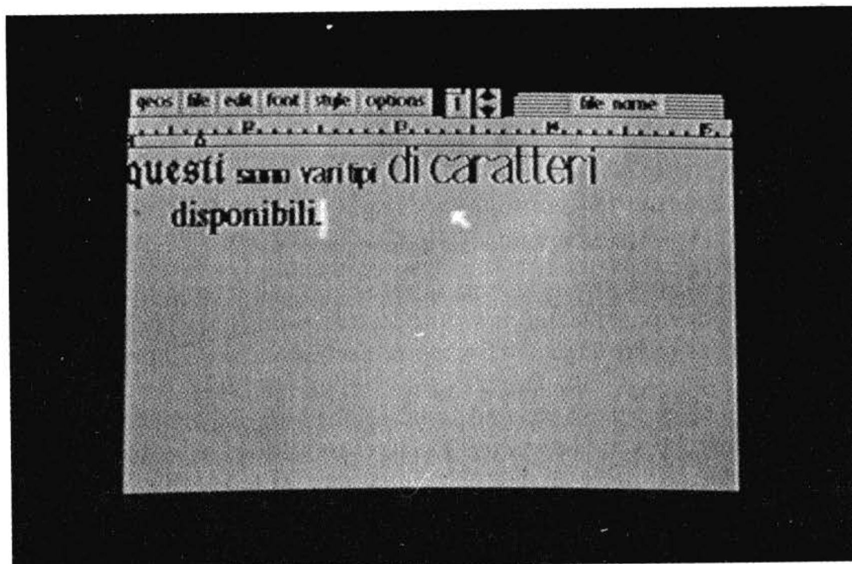


Figura 3.1 - GEOWRITE, il word processor.

OPEN vi permette di accedere a un documento creato in precedenza

QUIT vi permette di tornare al Desktop

Potete quindi cominciare a creare il vostro primo documento selezionando l'opzione CREATE.
Il GEOS procederà quindi a chiedervi il nome che volete assegnare al testo: sceglietene uno qualunque e inseritelo non dimenticando di schiacciare il tasto di RETURN alla fine.
A questo punto apparirà sul vostro schermo un foglio in bianco su cui potrete scrivere ciò che vorrete (fig. 3.1). In realtà sul video vi è rappresentata solo una piccola parte della pagina che avete a disposizione: questa è più larga e più lunga. Per avere un'idea della situazione osservate il riquadro in alto al centro (fig. 3.2): rappresenta l'intera pagina a disposizione.
Il rettangolino più scuro all'interno indica la zona visualizzata. Questo può essere spostato a piacere: - posizionate la freccia sul rettangolino e premete il tasto di fire - a questo punto potrete trascinarlo muovendo la manopola del joystick, e depositarlo sulla zona da visualizzare premendo nuovamente il tasto di fire. Il numero all'interno del riquadro indica la pagina su cui siete posizionati.

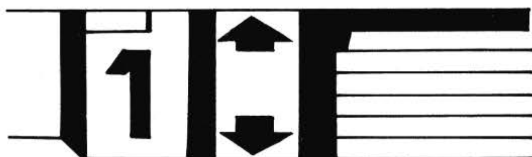


Figura 3.2 - Particolare della pagina di lavoro.

Un'altro modo per spostare la zona rappresentata all'interno di una pagina, consiste nel posizionare la freccia su uno dei due triangolini situati a destra del riquadro e di azionare il tasto di fire: potrete così avanzare o indietreggiare di una linea.
Sulla riga inferiore si trova una scala graduata da 1 a 7 che, oltre ad essere utile per fissare margini e tabulatori, vi consente di stabilire la posizione orizzontale di un qualsiasi carattere all'interno di una linea di testo. Provate quindi a scrivere una riga molto lunga (senza andare a capo): se all'inizio veniva visualizzata la zona dal 1 al 5, a un certo punto vi verrà mostrata la seconda parte della pagina dal 4 al 7 (sono numeri leggibili sulla scala graduata). Per tornare a visualizzare la parte sinistra potete:

— andare a capo, premendo RETURN
— utilizzare il riquadro visto prima spostando il rettangolino nella posizione voluta
— muovere mediante il joystick la freccia, in modo da cercare di superare il limite sinistro della zona visualizzata; vedrete così apparire la prima parte della vostra riga.

Gli ultimi due metodi sono applicabili anche per effettuare l'operazione inversa. Per il terzo bisogna però dirigere la freccia verso destra. Questi cambiamenti dello schermo possono disturbare chi scrive; percui si usa fissare un margine destro sulla tacca 5. Al momento della stampa, lo sposterete secondo le vostre esigenze.

Per fissare il margine destro procedete così:

— spostatevi sulla seconda metà della pagina in modo da visualizzare la tacca 7
— sotto di questa si trova un indice che segnala la posizione del margine
— mediante la manopola del joystick, posizionate la freccia sopra tale indice e premete il tasto di fire - ora potrete spostare il margine orizzontalmente
— fissate il margine sulla tacca desiderata premendo il tasto di fire.

Per spostare il margine sinistro procedete nello stesso modo, spostando però l'indice situato sotto la tacca 1. I margini definitivi vengono fissati una volta scritto e corretto il testo e dipendono dal formato della carta su cui scrivere. Il formato massimo è quello A4 verticale (modulo continuo 234 mm per 12", 72 linee per foglio).
In ogni caso ricordatevi che il GEOS provvede già a lasciare dello spazio libero dai limiti destro, sinistro, inferiore e superiore della pagina. Dimensioni diverse richiedono uno spostamento dei margini e l'inserimento di cambi di pagina (page breaks vedi il capitolo seguente) in modo che la stampa non esca dal foglio. Noi vi consigliamo sempre di fare dei tentativi in modo da ottenere la stampa migliore.

Ulteriori particolarità in fase di scrittura Il cursore è rappresentato da una linea verticale lampeggiante. Ogni tasto premuto genera un carattere che viene inserito nel testo nel luogo dove si trova il cursore; questo viene fatto avanzare di un posto.
Il fatto che voi inseriate caratteri significa che non cancellerete mai nulla sul vostro testo semplicemente scrivendo delle lettere, come avveniva quando scrivevate programmi in BASIC. Questa è una caratteristica molto utile comune a tutti i wordprocessors e imparerete ad apprezzarla con il tempo.

Per cancellare dei caratteri non vi rimane che usare il tasto di DELETE (DEL in alto a destra sulla tastiera) che eliminirà quello situato a sinistra del cursore; diversamente si può attivare un pezzo di testo e toglierlo seguendo le opzioni che vedremo però in seguito.
Un'altra caratteristica interessante di GEOWRITE è quella del wordwrap; questa fa in modo che i margini non interrompano a metà le parole che voi avete scritto. Infatti nel caso in cui la parola non possa essere contenuta sulla riga corrente, viene presa e spostata su quella seguente.
Inoltre come avrete notato il cambio di linea risulta essere automatico. Per imporne uno e sufficiente premere il tasto RETURN; è importante notare che questo è visto come un normale carattere ed è quindi eliminabile usando il tasto di DELETE. Provate ad esempio a scrivere la frase seguente: "Questo è un esempio di come" (RETURN) "ricongiungere due linee"
Ora prendete in mano in joystick e puntate la freccia sulla prima "r" di "ricongiungere": premete il tasto di fire e vedrete che il cursore si sarà spostato proprio nel punto indicato dalla freccia. Se ora provate a inserire degli spazi vedrete che questi sposteranno la linea inferiore verso destra; ora premete il tasto DELETE in modo da far tornare la parola "ricongiungere" contro il margine sinistro.
A questo punto premendo ancora una volta il tasto DELETE ottenerrete l'eliminazione del carattere RETURN e le due linee si ricongiungeranno. In pratica tale carattere è situato immaginariamente subito dopo l'ultimo della linea precedente: ben si comprende dunque il significato dell'operazione vista sopra.

Inoltre avete potuto vedere come sia possibile posizionare il cursore mediante il joystick. A tale proposito provate a scrivere le linee: "Gli spazi qui a destra" (RETURN) "non sono veri caratteri, bensì costituiscono una zona vuota" Cercate ora di posizionare il cursore sulla prima riga, nella zona senza caratteri a destra della parola "destra".
Non ci riuscirete: esso infatti si piazzerà sempre subito dopo la "a". La ragione di tale comportamento è che fra "destra" e "non" c'è un solo carattere e cioè RETURN: gli spazi rimasti sulla destra NON sono caratteri ma zone vuote.
Per cui ogni volta che cercherete di posizionare il cursore in una tale regione, ricordatevi che esso si metterà sempre subito dopo il primo carattere che incontrerà viaggiando immaginariamente verso sinistra (e verso l'alto).
Quanto visto fin qui ci aiuta a capire come i wordprocessors memorizzino i testi. Questi sono costituiti da una lunga sequenza di caratteri: alcuni alfanumerici (1,2,3,...,a,b,c,.....) e altri di controllo (RETURN per esempio). Al momento di scrivere il testo su schermo, il programma stampa tutti i caratteri in sequenza, facendo attenzione a rispettare i margini imposti: in definitiva

questi non modificano la struttura del file memorizzato.
Se scrivete molte linee, arriverete a un punto in cui la pagina verrà cambiata automaticamente. Un tale cambio può essere imposto selezionando l'opzione "page break" dal menu "options". In tal modo il wordprocessor inserisce nel testo un carattere di controllo che impone il cambio della pagina. Questo può essere cancellato con le stesse modalità viste per RETURN, riunendo così due sezioni di testo poste su due fogli separati ma contigui. Supponiamo allora di essere in pagina 1:

— selezionate dal menu superiore l'opzione "page break"
— dopo alcuni secondi appare la pagina 2 (vuota);
— scrivete alcuni caratteri (qualunque)
— posizionate il cursore sul primo carattere della pagina 2, in modo che sia in alto a sinistra attaccato al margine
— premete il tasto di DELETE
— dopo alcuni secondi apparirà una finestra supplementare che vi chiederà se cancellare l'ultimo carattere della pagina precedente, in questo caso quello di controllo "page break": voi selezionate la scelta O.K.
— in tal modo avrete riattaccato la pagina 2 alla 1

Bisogna infine dire che "page break" inserisce un foglio nuovo nel testo, incrementando di uno la numerazione degli eventuali fogli seguenti. Per chiudere il testo di prova che abbiamo creato e tornare al Desktop, selezionate l'opzione "quit" dal menu superiore "file"; sfogliando il taccuino del direttorio troverete una nuova icona, quella del testo che avete appena scritto. Per tornare a modificarlo, è sufficiente effettuarvi un doppio click sopra.
Per cancellarlo, trascinatelo sul contenitore delle immondizie.

3.3 FONT: come cambiare caratteri

Esaminiamo ora una delle caratteristiche più interessanti di GEOWRITE, vale a dire la possibilità di utilizzare più tipi di caratteri, di diversa dimensione nello stesso testo. Per accedere a tale possibilità bisogna attivare il menu "font". Comparirà una lista di tipi di caratteri (font) disponibili; questi dipendono dai file font memorizzati sul dischetto in uso (sul taccuino del direttorio sono rappresentati da icone contenenti la scritta "FONT"). Noterete che un asterisco indica il tipo di stampa attualmente in uso. Per sceglierne uno nuovo selezionate il nome che vi interessa; successivamente apparirà un altro menu da cui dovrete scegliere l'altezza in numero di punti del carattere. Per modificare solo quest'ultima è sufficiente, dopo avere attivato il medesi-

mo menu, selezionare il font in uso, e in seguito modificarne le dimensioni a piacere. La scelta della spaziatura fra caratteri e fra linee successive è automatica, e fa sempre sì che non vi siano sovrapposizioni.

3.4 Gli stili di scrittura

Una volta scelto il font più adatto, può rendersi necessario in certi punti del testo un cambiamento dello stile di scrittura; per fare ciò attivate il menu "style". L'asterisco indica quello attualmente in uso; se si trova a sinistra di "plain text" vuol dire che la scrittura è normale. Selezionate lo stile voluto nel solito modo; la differenza con gli altri menu superiori, consiste nel fatto che più stili possono essere contemporaneamente attivi. Potrete così scrivere,per esempio, in grassetto e sottolineato (bold e underlined). Per eliminarne uno, attivate il menu superiore "style", puntate la freccia sul nome relativo e premete il pulsante di fire. Riattivando il menu noterete la sparizione dell'asterisco dal fianco dello stile disabilitato.
Ogni volta che posizionate il cursore mediante il joystick in una zona di testo, il font e lo stile con cui scriverete vengono determinati dal carattere precedente. Per cambiare bisogna quindi attivare i menu "font" e "style" e scegliere le opzioni che interessano.

3.5 Attivazione di zone di testo

Un particolare uso del joystick e del cursore, riguarda l'attivazione di sezioni del testo. Per compiere tale operazione è sufficiente: - posizionare il cursore sul primo carattere da attivare - premere il tasto di fire e non rilasciarlo - muovere la manopola del joystick in modo da attivare una parte del testo visualizzato; potete muovervi in tutte le direzioni ma non dimenticate di tenere schicciato il pulsante di fire - rilasciare il tasto di fire quando avrete attivato la zona voluta La parte di testo così attivata viene messa in evidenza tramite lo scambio del colore di fondo con quello dei caratteri (reverse).
A questo punto le operazioni possibili sono le seguenti.
Cancellazione: È sufficiente premere il tasto di DELETE (INST/DEL) Sostituzione: Se dovete sostituire la sezione attivata con un testo diverso, scrivetelo subito: GEOWRITE provvederà a cancellare il testo vecchio sostituendolo con quello nuovo Cambiamento dei font e dello stile: Potete cambiare il tipo e lo stile di tutti i caratteri attivati, selezionando le opzioni desiderate dai menu superiori "font" e "style" (vedi paragrafi prcedenti) Trasferimento in text

scrap: Nella sezione dedicata a GEOPAINT abbiamo visto l'uso di photo scrap; un file molto simile esiste pure per i testi e si chiama text scrap.
Per trasferirvi il blocco che è stato attivato è sufficiente selezionare dal menu "edit" le opzioni:

— CUT , se si intende togliere il blocco per trasferirlo altrove nel testo
— COPY , se si vuole copiarlo Una volta copiato in text scrap, il testo può essere incollato ovunque nel nostro documento.

È sufficiente posizionare il cursore nel punto dove dovrà essere inserito e selezionare dal menu "edit" l'opzione "paste". A questo punto comparirà un altro menu che vi chiederà se incollare delle immagini (GEOPAINT) o del testo: selezionate quest'ultimo (opzione "text").

Questa operazione di inserimento non svuota il contenuto di text scrap e quindi può essere ripetuta più volte. Da notare che i caratteri vengono trasferiti senza essere modificati. Se aveste in precedenza scelto l'opzione "picture", avreste copiato nel vostro documento l'immagine contenuta in photo scrap. In tal modo è possibile combinare testi e disegni, per documenti estremamente professionali. Siccome text scrap può contenere un blocco di testo alla volta, si è sentita la necessità di creare un raccoglitore di tali parti di testo: il suo nome è text album e il suo uso sarà approfondito nella sezione dedicata agli accessori del GEOS.

3.6 Visualizzazione delle pagine

Una volta creato un documento a più fogli diventa utile il poter passare semplicemente da uno all'altro. Per fare ciò è sufficiente attivare il menu "options" e selezionare l'opzione voluta. LAST PAGE: visualizza la pagina precedente NEXT PAGE: visualizza la pagina seguente GOTO PAGE: il GEOS vi chiederà il numero della pagina da visualizzare; inseritelo non dimenticando di premere RETURN alla fine HIDE PICTURES : se avete inserito immagini, questa opzione non le farà apparire, aumentando la velocità di visualizzazione SHOW PICTURES : le immagini vengono di nuvo mostrate PAGE BREAK : inserisce un cambio pagina

3.7 Gestione dei file

Avrete sicuramente notato che mentre scrivete un testo il disco non rimane sempre inattivo e anzi risulta essere molto spesso utilizzato. Questo succede in

quanto GEOPAINT tiene nella memoria centrale solo la parte visualizzata, e ogni volta che questa cambia, il file viene aggiornato. Ciò può essere esplicitamente richiesto selezionando "UPDATE FILE" dal menu "file". In tal modo sarete sicuri che anche le ultime modifiche sono state salvate su disco, e potrete spegnere il COMMODORE 64 senza preoccupazioni.
Esiste poi l'opzione inversa, che vi permette di ripristinare la zona visualizzata in caso che abbiate fatto delle modifiche non volute; questa è "recover from file" sempre presente nel menu "file". Da notare che tale operazione ha senso se e solo se il contenuto del file non è stato aggiornato già automaticamente da GEOWRITE (basta osservare il drive: ogni volta che funziona viene operata una modifica dei testi su disco).
Nel menu "file" sono poi disponibili altre opzioni: CLOSE : aggiorna il documento con le ultime modifiche e va al menu iniziale di GEOWRITE PRINT : stampa il documento PREVIEW PAGE : visualizza l'aspetto finale della pagina su cui si sta lavorando; per tornare al testo posizionatevi su O.K. e premete il tasto di fire QUIT : aggiorna il documento con le ultime modifiche e va al Desktop

3.8 Uso dei tabulatori

L'uso dei tabulatori permette di raggiungere una predefinita posizione orizzontale all'interno del testo, premendo contemporaneamente i tasti "CTRL" e "I". Il tabulatore definisce tale punto; per piazzarne uno, seguite la procedura seguente. - usando il joystick posizionate la freccia fra la scala graduata e il riquadro del testo nel punto desiderato - premete il tasto di fire e vedrete apparire un indicatore che segnala la presenza del tabulatore.
Per eliminare un tabulatore, puntate la freccia sopra l'indicatore relativo, e dopo aver premuto il tasto di fire, trascinatelo sul margine sinistro; qui rilasciatelo premendo ancora il pulsante di fire.

CAPITOLO 4

GLI ACCESSORI DI GEOS

I programmi che sono disponibili all'interno di GEOS appartengono a due classi: le APPLICAZIONI e gli ACCESSORI.
Questi ultimi si differenzano dalle APPLICAZIONI perchè sono utilizzabili in qualsiasi momento. Ciò vuol dire che è possibile utilizzare l'accessorio CALCULATOR anche se abbiamo attivata (cioè stiamo eseguendo) l'applicazione GEOPAINT.
Non possiamo invece utilizzare GEOWRITER durante l'esecuzione di GEOPAINT perchè sono entrambe delle applicazioni.
Per utilizzare un qualsiasi accessorio è sufficiente selezionarlo dal pull-down menu GEOS che è sempre presente sullo schermo.
Se proviamo ad attivare da DESKTOP il pull-down menu potrete scegliere fra i seguenti accessori:
GEOS INFO
DESKTOP INFO
CHOSE PRINTER
PREFERENCE MGR
NOTE PAD
PHOTO MANAGER
TEXT MANAGER
ALARM CLOCK
CALCULATOR
In realtà queste non sono tutti degli accessori veri e propri. Infatti GEOS INFO e DESKTOP INFO, se attivati, presentano su video delle informazioni sul sistema operativo GEOS e su DESKTOP.
Per disattivarli è sufficiente premere il tasto di fire. Anche CHOOSE PRINTER non può essere considerato un accessorio vero e proprio poichè serve solo a mostrare il DRIVER DI STAMPA correntemente selezionato. Questo è un programma che vi permette di stampare i vostri documenti, e dipende dal tipo di apparecchio usato.
Sul vostro dischetto ce ne sono cinque rappresentati con icone che mostrano una stampante. Viene automaticamente selezionato il primo DRIVER incontrato sul taccuino del direttorio; dovrete quindi spostare quello relativo alla vostra stampante, in testa agli altri. Esaminiamo ora uno ad uno i veri accessori di GEOS:

PREFERENCE MGR

Questo accessorio, a cui si accede selezionando dal pull-down menu l'opzione corrispondente, permette di definire alcuni parametri generali per il geos. Fra questi troviamo il colore, la forma ed i parametri che definiscono la rapidità dei movimenti della freccia controllata dal joystick, il colore del fondo, il colore dei caratteri, ecc.

Quando l'accessorio PREFERENCE MGR viene aperto si presenta una schermata identica a quella riportata in fig. 4.1. Su questa vengono rappresentati i PARAMETRI GEOS accessibili dall'utente raggruppati in diverse AREE FUNZIONALI. Ciascuna di queste verrà ora esaminata in dettaglio.

CONTROL OPTIONS:

Selezionando EXIT si chiude l'accessorio PREFERENCE MGR; con LOAD si richiamano da disco i precedenti valori dei PARAMETRI GEOS mentre SAVE registra quelli attuali.

CHANGE

serve per rendere operativi tutti i cambiamenti effettuati e DEFAULT provvede ad assegnare dei valori predefiniti ad ogni campo.

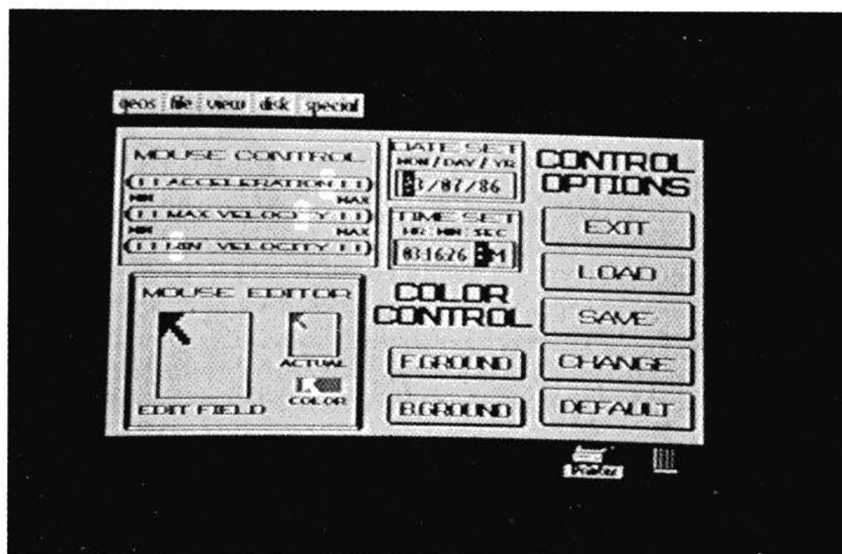


Figura 4.1 - L'accessorio PREFERENCE MGR.

MOUSE CONTROL:
il sistema operativo GEOS, per facilitare l'uso della freccia controllata dal joystick, ci permette di variarne la velocità massima, minima e l'accelerazione. I loro valori sono indicati da un indice posto su ciascuna delle scale contenute in MOUSE CONTROL.
Posizionandovi sopra la freccia e premendo il tasto di fire, potrete variarne la posizione. Premete il tasto di fire di nuovo per fissarlo nel punto desiderato.

MOUSE EDIT:
Sempre per rendere gradevole l'interazione con il GEOS, viene data all'utente la possibilità di definirsi la forma ed il colore dell'indicatore controllato dal joystick usualmente rappresentato con una freccia. Per agire sul colore bisogna posizionarsi all'interno della finestra denominata ACTUAL COLOR e premere il tasto di fire. Per agire sulla forma bisognerà portarsi all'interno di EDIT FIELD.
Qui è possibile disegnare o cancellare ogni singolo pixel che costituisce la sagoma della freccia premendo in corrispondenza del pixel desiderato il tasto di fire.

DATE SET:
Su quest'area funzionale si agisce per modificare la data corrente seguendo questi passi:

a) portare il la freccia all'interno di DATE SET.

b) editare la nuova data da tastiera e terminarla con il tasto RETURN.

La barra spaziatrice ci permette di selezionare una particolare cifra.

TIME SET:
In quest'area sono contenuti i parametri per l'ora corrente. Per modificarli si dovrà procedere così:

a) portare la freccia all'interno di quest'area funzionale.
b) digitare a oppure p (a = antimeridiamo p = pomeridiano).
c) introdurre l'ora corrente terminandola con il tasto RETURN.

L'uso della barra spaziatrice è identico a quanto detto per DATE SET.
CONTROL COLOR:
F.GROUND e B.GROUND servono per scegliere rispettivamente il colore dei caratteri ed il colore dello sfondo. Per selezionare nuovi colori è sufficiente premere il tasto di fire all'interno di queste aree.

NOTE PAD

Questo accessorio costituisce un taccuino sempre disponibile per registarvi annotazioni di ogni genere. Per aprirlo è sufficiente attivare dal pull-down menu l'opzione corrispondente. Vi si presenterà dopo qualche istante una schermata simile a quella riportata in fig. 4.2.

L'uso di NOTE PAD è semplicissimo: per inserire un appunto è sufficiente digitarlo da tasiera mentere per correggerlo o toglierlo si ricorre al tasto INST DEL. Per voltare le pagine ci si posiziona sull'angolo in basso a sinistra del taccuino e si preme il tasto di fire. Analogamente, per chiudere NOTE PAD, ci si posiziona sul tasto posto in alto a destra e si preme il tasto di fire.

PHOTO MANAGER

In GEOPAINT, quando si è illustrato l'uso del pull-down menu EDIT, si è parlato di un file di sistema detto PHOTO SCRAP. Su questo si poteva memorizzare un'immagine. Ciò risultava di grande utilità poichè consentiva di muovere, copiare e cancellare dei particolari di un disegno. Tuttavia in PHOTO SCRAP non era possibile salvare più di un'immagine per volta e ciò impediva di realizzare un'archivio grafico.

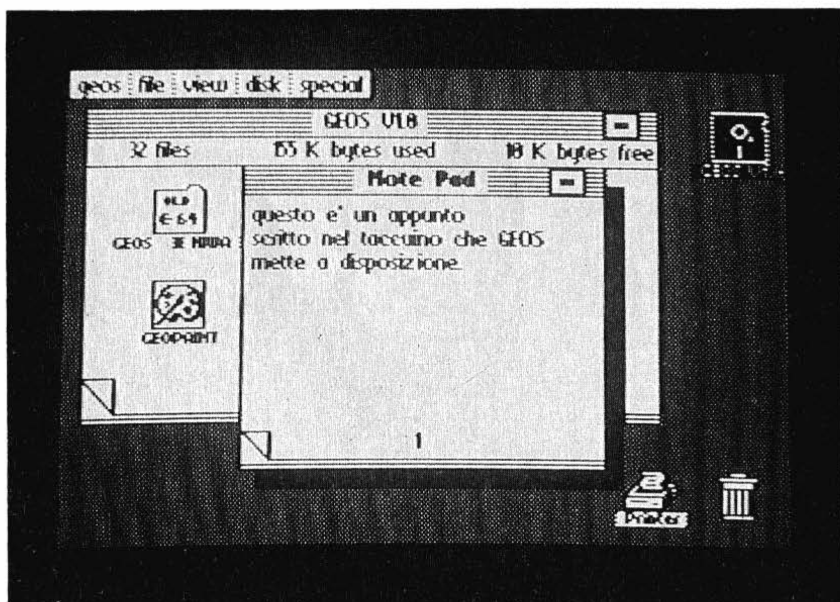


Figura 4.2 - L'accessorio NOTE PAD.

Con PHOTO MANAGER questo problema è superato. Infatti questo accessorio è un vero e proprio gestore dell'archivio costituito da PHOTO ALBUM. Le tre operazioni che PHOTO MANAGER consentedi compiere sono:

1) AGGIUNTA DEL DISEGNO CONTENUTO IN PHOTO SCRAP AL PHOTO ALBUM
2) COPIARE UN DISEGNO DAL PHOTO ALBUM AL PHOTO SCRAP
3) TOGLIERE UN DISEGNO DAL PHOTO ALBUM TRASFERENDOLO IN PHOTO SCRAP

Veniamo ora al dettaglio di queste due operazioni. Dopo aver aperto PHOTO MANAGER vi si presenterà una videata simile a quella di fig. 4.3. Per copiare il contenuto di PHOTO SCRAP in PHOTO ALBUM sarà sufficiente selezionare dal pull-down menu EDIT l'opzione PASTE. Diversamente se intendiamo copiare su PHOTO SCRAP l'immagine che PHOTO MANAGER ci sta mostrando sul video si dovrà attivare dal pull-down menu EDIT o l'opzione CUT oppure l'opzione COPY.

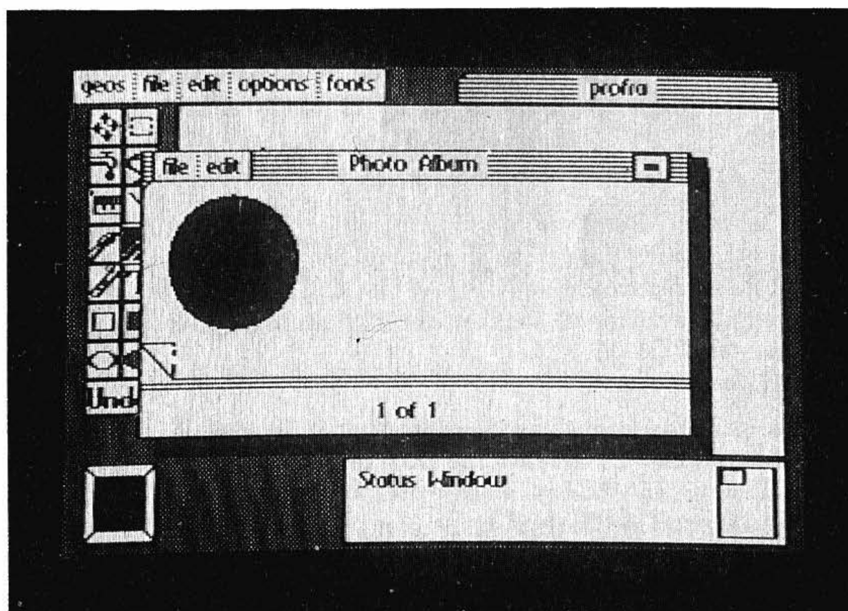


Figura 4.3 - L'accessorio PHOTO ALBUM.

A differenza di COPY, CUT provvede anche a toglire da PHOTO ALBUM l'immagine che viene copiata in PHOTO SCRAP. Se desideriamo sfogliare PHOTO ALBUM dovremo clickare sull'angolo in basso a sinistra del riquadro che PHOTO MANAGER utilizza per presentarci le immagini di PHOTO ALBUM.

Per chiudere questa applicazione potremo indifferentemente posizionarci sul tasto posto in alto a destra nella finestra in cui PHOTO MANAGER opera e premere il tasto di fire oppure selezionare dal pull-down menu FILE l'opzione CLOSE.

TEXT MANAGER

Il modo di operare di questo accessorio è quasi identico a quello di PHOTO MANAGER e pertanto indicheremo soltato le differenze che vi si possono riscontrare. Innanzi tutto i file su cui TEXT MANAGER lavora sono TEXT ALBUM e TEXT SCRAP.

Questi, pur svolgendo ruoli assolutamente analoghi a quelli di PHOTO ALBUM e PHOTO SCRAP, se ne differenziano perchè non sono adatti a trattare disegni ma soltanto testi, intesi secondo la definizione più sopra riportata. La seconda differenza fra queste due applicazioni è costituita dalla schermata che si presenta quando si apre TEXT MANAGER. Questa, pur essendo quasi identica a quella usata da PHOTO MANAGER, offre in più la possibilità di scorrere il testo utilizzando le due frecce verticali poste sul margine inferiore del riquadro in cui TEXT MANAGER mostra TEXT ALBUM.

ALARM CLOCK

Questo simpatico accessorio funziona in maniera simile ad un orologio da polso che ci mostra l'ora e che dispone di una sveglia. Questa una volta predisposta suonerà anche se ALARM CLOCK non è aperto. In fig. 4.4 è stata riportata la schermata che si ottiene selezionando questo accessorio dal pull-down menu GEOS.

Vi sono indicati con:

1) il visore da cui si legge l'ora e su cui si imposta la sveglia.
2) il pulsante/spia che permette di selezionare la modalità di funzionamento OROLOGIO o SVEGLIA, l'una indicata con il simbolo di un orologino l'altra con una CAMPANELLA.
3) il pulsante che attiva o disattiva la sveglia. Questo tasto agisce agisce solo in modalità sveglia.
4) il tasto che ci permette di chiudere questa applicazione.
5) la spia che ci avverte dell'inserimento della sveglia.

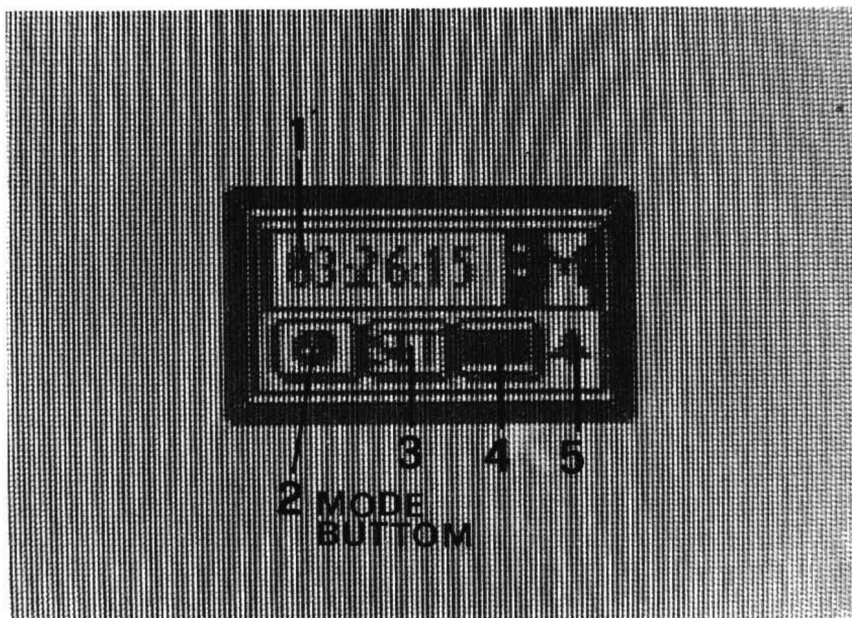


Figura 4.4 - L'accessorio ALARM CLOCK.

Desiderando impostare l'ora bisogna procedere così:

— accertarsi di essere in MODALITÀ OROLOGIO, osservando il simbolo mostrato dal PULSANTE/SPIA; se non foste in tale modalità, posizionatevi sopra di questo e premete il tasto di fire.
— digitare il tasto A oppure P se si intende inserire un'ora antimeridiana oppure pomeridiana
— digitare ora, minuti e secondi
— posizionatevi sul pulsante di set e premere il tasto fire.

Desiderando invece impostare l'ora di sveglia si dovranno eseguire queste altre operazioni:

— accertarsi di essere in modalità SVEGLIA, osservando il simbolo mostrato dal PULSANTE/SPIA; se non foste in tale modalità, posizionatevi sopra di questo e premete il tasto di fire.
— digitare il tasto A oppure P se si intende inserire un'ora antimeridiana oppure pomeridiana
— digitare ora, minuti e secondi

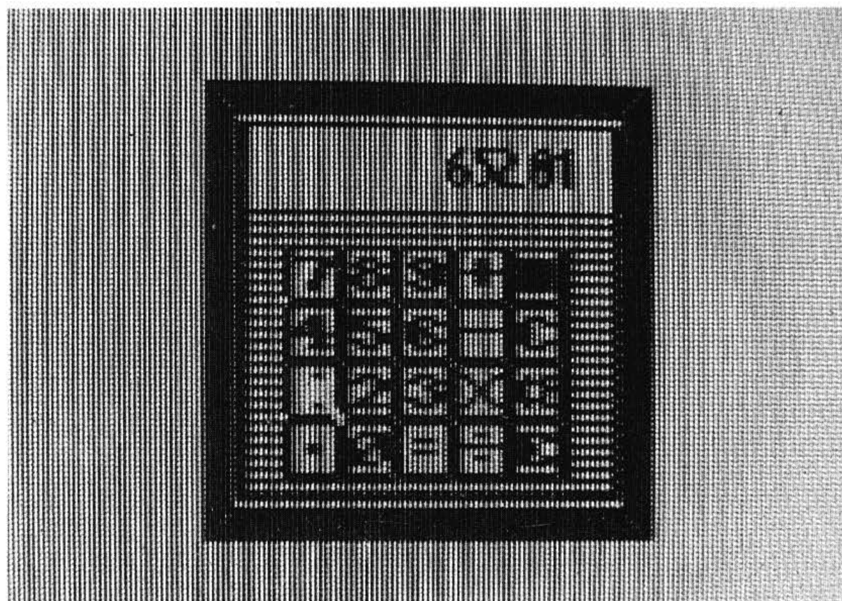


Figura 4.5 - L'accessorio CALCULATOR.

— posizionatevi sul pulsante di set oppure sul pulsante mode e premete il tasto di fire.

CALCULATOR

Anche questo accessorio è di uso semplicissimo. Una volta aperto vi si presenterà una schermata analoga a quella riportata in fig. 4.5. I tasti sono quelli consueti di una normale calcolatrice da tasca e sono azionabili posizionandovisi sopra e premendo il fire. Il tasto di OFF, posto in alto a destra, serve per chiudere questo accessorio. Anche nell'uso non vi sono differenze rispetto alle ormai diffuse calcolatrici tascabili.

CAPITOLO 5

ALCUNI CONSIGLI FINALI

5.1 CREAZIONE DI UN WORKDISK

Siccome il vostro dischetto GEOS risulta essere quasi interamente occupato, diventa necessario creare un disco di lavoro su cui siano presenti solo i file indispensabili. A tale proposito tratteremo l'esempio di un workdisk dedicato al wordprocessor.

— Formattate un disco con il GEOS; il nome può essere qualunque
— Copiatevi sopra dal disco sistema i file che ritenete necessari

Desktop: non è fondamentale, ma è più comodo averlo su ogni disco
Geowrite: è l'applicazione cui è dedicato questo disco di lavoro
Font:copiate i file di font che ritenete necessari; potrete sempre comunque aggiungerne uno
Stampante: copiate un solo file (Printer driver), quello relativo alla vostra stampante (per es. Commodore)
Text manager: copiatelo solo se volete usare il text album; quest'ultimo può essere copiato da altri dischetti, ma su ognuno non ne possono coesistere due
Photo Manager: copiatelo insieme al photo album, se e solo se intendete incollare immagini nel vostro testo

Ricordate che meno file copiate più spazio avrete a disposizione per i vostri documenti. Questo disco NON può essere usato per caricare il sistema GEOS: per compiere tale operazione dovrete usare sempre il disco di sistema.
A volte può essere noioso il dover sfogliare le pagine del taccuino del direttorio per aprire il documento su cui si sta lavorando. Allora è utile applicare un piccolo trucco, consistente nel trascinare l'icona del file sul fondo, rilasciandola sotto il taccuino, in modo che sia sempre visibile indipendentemente dal foglio visualizzato. Il GEOS si ricorderà di tale operazione e ogni volta che aprirete il disco di lavoro vedrete comparire, in basso e separato, il file. In tale zona avete posto per quattro icone. Per aprire tale file sarà sufficiente effettuare il solito doppio click; potrete poi riportarlo nel taccuino quando vorrete.

5.2 Come legare i vostri programmi Basic al GEOS

Abbiamo visto in un capitolo precedente come fosse possibile lanciare i vostri programmi basic dal GEOS. Con un semplice accorgimento è pure possibile fare in modo che questi, quando siano terminati, vi riportino nel GEOS. I vostri programmi dovrebbero terminare con la parola chiave END; al posto di questa inserite le linee basic:

1000 print "Inserite il disco di sistema del GEOS." 1010 print "Quando avete terminato premete un 1020 print "tasto qualunque." 1030 get a$:if a$ = "" then 1030 1040 load "geos",8,1

In tal modo ritornerete in GEOS.

Il testo è rivolto ai nuovi acquirenti del C64 che necessitano di una conoscenza approfondita sia del linguaggio Basic residente, sia del nuovo sistema operativo Geos con cui la Commodore ha recentemente rilanciato il C64. Infatti questo sistema permette di trasformare il C64 in un macchina particolarmente studiata sulle esigenze dell'utente, il cui uso non richiede alcuna conoscenza informatica preliminare, essendo stato progettato per risultare pratico e immediato. Tuttavia la disponibilità di pacchetti software di cosi semplice utilizzo, non rende inutile la conoscenza del Basic che, restando uno dei più usati linguaggi di programmazione, risulta essere tuttora uno strumento insostituibile per comunicare con il calcolatore.

Il libro risulta diviso, data la diversa peculiarità degli argomenti, in due parti nettamente distinte tra loro, tali da rendere la loro lettura indipendente l'una con l'altra.

La prima, relativa al linguaggio Basic, affronta le problematiche tipiche della programmazione in modo da far maturare nel lettore la sensibilità necessaria a stendere un programma che vada aldilà del banale. Per questo, oltre a spiegare minuziosamente il significato delle varie istruzioni Basic, si è fatto ricorso ad un gran numero di esempi che forniscono al lettore gli strumenti per ricorrere, di volta in volta, alle istruzioni più adeguate per risolvere un dato problema. Ogni unità del corso è completata da interessanti esercizi che permettono di valutare l'apprendimento conseguito.

La seconda parte è interamente dedicata all'illustrazione del Geos e delle sue applicazioni GeoWrite e GeoPaint, applicazioni interessantissime per la facilità con cui è possibile redarre documenti o disegni. Viene trattato nel dettaglio l'uso dei vari strumenti disponibili ponendo sempre l'accento sugli aspetti che possono risultare meno chiari o immediati.

L. 45.000 Cod. 548B ISBN 88-7056-255-7